Final Project Submission

Please fill out:

- Student name: Jessica Miles
- Student pace: full time
- Scheduled project review date/time: 6/23: 4 PM
- Instructor name: James Irving

INTRODUCTION

Business Problem

Social media presence is an important part of a modern brand's marketing strategy. But these platforms not only allow brands to broadcast their messages directly to consumers, they also allow consumers to voice their feedback and opinions about the brands candidly, and in a public forum. This translates to a responsibility on one hand, but also an opportunity on the other hand, for companies to listen to and respond to feedback from customers.

Many companies use the Net Promoter Score (NPS) as a measure of customer satisfaction and loyalty. However, NPS survey response rates can be low (15%-20% is considered decent) and non-response bias makes the resulting scores unreliable. NPS is also usually just a single question asking how likely a customer is to recommend the company's product to someone else; they may not provide the mechanism for the respondent to give specific feedback about what lead to their answer. Analysis of other channels where customers provide feedback, such as Twitter, could supplement frequently sparse NPS data.

If technology could take a first pass on determining the sentiment of tweets, large companies would have a better chance at winnowing constructive, actionable feedback from trolling or irrelevant comments.

Questions to consider:

- What are the business's pain points related to this project?
- How did you pick the data analysis question(s) that you did?
- Why are these questions important from a business perspective?

When two brands are in competition with each other for business, each seeks to understand how consumers feel about their products and company versus others. Knowing what customers like best about your products and company can inform strategies to keep those best-loved qualities around, as well as understanding what your niches are. Likewise, knowing what customers do not like about your products will illuminate areas where change may be needed, especially if your competitor does well in those areas.

Customers may provide feedback directly to you, but the ability to analyze what they say to others on public platforms such as Twitter may provide helpful insights from people who would not normally reach out.

In this analysis, I will build a model to predict the positive or negative sentiment related to different technology products and brands which are in competition. The goal is to provide information to both companies regarding what customers like and dislike most about their products and brands.

I will use a dataset consisting of tweets that appear to have been collected during a SXSW event. Both Apple and Google had corporate presences in the forms of talks and pop-up stores, as well as apps that attendees used during the event.

The tweets were coded by humans, who were asked to classify them based on emotion related to brands and products. Here is the brief overview from [data.world]: (https://data.world/crowdflower/brands-and-product-emotions)

> Contributors evaluated tweets about multiple brands and products. The crowd was asked if the tweet expressed positive, negative, or no emotion towards a brand and/or product. If some emotion was expressed they were also asked to say which brand or product was the target of that emotion.

I will focus on

OBTAIN

Data Understanding

This data comes from CrowdFlower. It consists of a corpus of about 9,000 tweets which humans were asked to label according to whether they were related to a particular brand or product, and whether a positive, negative, or no emotion was expressed. Tweets about brands or products were labeled with the specific brand or product.

In [281…

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re
import html
import string
import joblib
from chardet.universaldetector import UniversalDetector

import nltk
from nltk.probability import FreqDist
from nltk.tokenize import TweetTokenizer, word_tokenize, wordpunct_tokenize
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
```

```
from wordcloud import WordCloud

import spacy

from sklearn import metrics
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_sc
from sklearn.dummy import DummyClassifier
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfTransformer, CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV

%matplotlib inline
```

In [282…
```
pd.set_option("display.max_colwidth",150)
```

In [283…
```
# try to detect chatacter encoding of the file
detector = UniversalDetector()

for line in open('data/judge-1377884607_tweet_product_company.csv', 'r+b').readl
    #print(line)
    detector.feed(line)
    if detector.done: break

detector.close()
print(detector.result)
```

```
{'encoding': 'Windows-1254', 'confidence': 0.43036719349968755, 'language': 'Tur
kish'}
```

That wasn't especially helpful. I tried cp1254 codec and it was not successful.

In [284…
```
# read in data. Had to switch to latin_1 encoding because encountered errors
# with default UTF-8 and windows 1254

df = pd.read_csv('data/judge-1377884607_tweet_product_company.csv',
                 encoding='latin_1')
df.head()
```

Out[284…

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_
0	.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.	iPhone	Nega
1	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW	iPad or iPhone App	Posi

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	iPad	Posi
3	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	iPad or iPhone App	Nega
4	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	Google	Posi

In [285…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9093 entries, 0 to 9092
Data columns (total 3 columns):
 #   Column                                              Non-Null Count  Dtype 
---  ------                                              --------------  ----- 
 0   tweet_text                                          9092 non-null   object
 1   emotion_in_tweet_is_directed_at                     3291 non-null   object
 2   is_there_an_emotion_directed_at_a_brand_or_product  9093 non-null   object
dtypes: object(3)
memory usage: 213.2+ KB
```

In [286…
```
# check out the one null in the tweet_text column
df.loc[df['tweet_text'].isna()]
```

Out[286…

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_or_proc
6	NaN	NaN	No emotion toward brand or proc

In [287…
```
# verified it's blank in the source CSV as well. Going to drop it.
df.dropna(subset=['tweet_text'], inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9092 entries, 0 to 9092
Data columns (total 3 columns):
 #   Column                                              Non-Null Count  Dtype 
---  ------                                              --------------  ----- 
 0   tweet_text                                          9092 non-null   object
 1   emotion_in_tweet_is_directed_at                     3291 non-null   object
 2   is_there_an_emotion_directed_at_a_brand_or_product  9092 non-null   object
dtypes: object(3)
memory usage: 284.1+ KB
```

In [288…
```
# rename the columns to be less verbose
```

```
col_dict = {'emotion_in_tweet_is_directed_at':'product',
            'is_there_an_emotion_directed_at_a_brand_or_product':'emotion'}
df.rename(columns=col_dict, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9092 entries, 0 to 9092
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   tweet_text  9092 non-null   object
 1   product     3291 non-null   object
 2   emotion     9092 non-null   object
dtypes: object(3)
memory usage: 284.1+ KB
```

In [289…
```
# check out classes
df['emotion'].value_counts()
```

Out[289…
```
No emotion toward brand or product    5388
Positive emotion                      2978
Negative emotion                       570
I can't tell                           156
Name: emotion, dtype: int64
```

In [290…
```
# what are the values in the product column? How do they match up to emotions?
df.groupby(by=['emotion', 'product'], dropna=False).count()
```

Out[290…

		tweet_text
emotion	**product**	
I can't tell	**Apple**	2
	Google	1
	Other Google product or service	1
	iPad	4
	iPhone	1
	NaN	147
Negative emotion	**Android**	8
	Android App	8
	Apple	95
	Google	68
	Other Apple product or service	2
	Other Google product or service	47
	iPad	125
	iPad or iPhone App	63
	iPhone	103
	NaN	51
No emotion toward brand or product	**Android**	1
	Android App	1

		tweet_text
emotion	product	
	Apple	21
	Google	15
	Other Apple product or service	1
	Other Google product or service	9
	iPad	24
	iPad or iPhone App	10
	iPhone	9
	NaN	5297
Positive emotion	Android	69
	Android App	72
	Apple	543
	Google	346
	Other Apple product or service	32
	Other Google product or service	236
	iPad	793
	iPad or iPhone App	397
	iPhone	184
	NaN	306

In [291…
```
# let's get a sample of "I can't tell"
df[df['emotion']=="I can't tell"]['tweet_text'][:10].values
```

Out[291…
```
array(['Thanks to @mention for publishing the news of @mention new medical Apps
at the #sxswi conf. blog {link} #sxsw #sxswh',
       '\x89ÛÏ@mention &quot;Apple has opened a pop-up store in Austin so the ne
rds in town for #SXSW can get their new iPads. {link} #wow',
       'Just what America needs. RT @mention Google to Launch Major New Social N
etwork Called Circles, Possibly Today {link} #sxsw',
       'The queue at the Apple Store in Austin is FOUR blocks long. Crazy stuff!
#sxsw',
       "Hope it's better than wave RT @mention Buzz is: Google's previewing a so
cial networking platform at #SXSW: {link}",
       'SYD #SXSW crew your iPhone extra juice pods have been procured.',
       'Why Barry Diller thinks iPad only content is nuts @mention #SXSW {lin
k}',
       'Gave into extreme temptation at #SXSW and bought an iPad 2... #impulse',
       'Catch 22\x89Û_ I mean iPad 2 at #SXSW : {link}',
       'Forgot my iPhone for #sxsw. Android only. Knife to a gun fight'],
      dtype=object)
```

Interesting. To me, these sample tweets labeled 'I can't tell' can all be classified pretty easily. However, they tend to be either neutral, or include nuanced sentiment such as sarcasm.

I'm definitely questioning the quality of the labeling at this point; if it was crowd sourced to a group of people I would have at least expected a second pass QC team to have gone through

the 'I can't tell' set and classified them. The fact that they did not may mean that there WAS no second pass QC step, in which case the consistency and quality of this coding may be mediocre.

In [292…
```
# for tweets with positive sentiment but no product listed, what are those about
df.loc[(df['emotion']=='Positive emotion') & (df['product'].isna()),
       ['tweet_text', 'emotion']]
```

Out[292…

	tweet_text	emotion
46	Hand-Held ‰Û÷Hobo‰Ûª: Drafthouse launches ‰Û÷Hobo With a Shotgun‰Ûª iPhone app #SXSW {link}	Positive emotion
112	Spark for #android is up for a #teamandroid award at #SXSW read about it here: {link}	Positive emotion
131	Does your #SmallBiz need reviews to play on Google Places...We got an App for that.. {link} #seo #sxsw	Positive emotion
157	@mention #SXSW LonelyPlanet Austin guide for #iPhone is free for a limited time {link} #lp #travel	Positive emotion
337	First day at sxsw. Fun final presentation on Google Doodles. #GoogleDoodle #sxsw	Positive emotion
...	...	...
8898	@mention What's the wait time lookin like? The Apple Store up north is already sold out, any word on the #SXSW inventory?	Positive emotion
9011	apparently the line to get an iPad at the #sxsw store grew by 2 blocks to 5 blocks in the past 30 mins. WUT.	Positive emotion
9049	@mention you can buy my used iPad and I'll pick one up tomorrow ;-) #sxsw	Positive emotion
9052	@mention You could buy a new iPad 2 tmrw at the Apple pop-up store at #sxsw: {link}	Positive emotion
9054	Guys, if you ever plan on attending #SXSW, you need 4 things, skinny jeans, flannel shirt, beard and an iPad #imanoutcast...	Positive emotion

306 rows × 2 columns

OK, I am also disagreeing with the coding on some of these as well.

Some I don't think have a positive emotion towards a product at all, such as rows 46, 131, 157, and 9054. Simply mentioning an app for a platform doesn't necessarily mean positive emotion was directed towards the app.

Others I can believe are slightly positive, but it's also clear which product or brand the emotion is directed towards, so I'm not sure why that is missing. Row 337 is definitely about a Google product that was showcased in a presentation, and 9049 is clearly related to an iPad.

SCRUB

Data Preparation

I performed some preprocessing on the tweets in doc form (i.e. before tokenizing). I performed these steps on all of the tweets regardless of class label, so that I can use any or all of them in my models later.

The main steps I did in this stage were:

- Checked for HTML tags (there were none)
- replaced HTML character codes with ASCII
- removed non-ASCII characters
- removed control characters
- Removed words that consisted of only numbers
- replaced URLs and links with a `{link}` placeholder

I considered doing some steps such as removing hashtags up front, but then I wouldn't be able to see the results of having them in. Instead, I copied the hashtags into a separate column so I could see what the most common ones were.

Tasks such as stopwords removal and handle (@mention) removal will be handled in preprocessing using the NLTK `TweetTokenizer` and sklearn `CountVectorizer` .

In [293…
```
# Make a copy of the tweet text, so I can keep the original pristine
df['cleaned'] = df['tweet_text']
df['cleaned'].head()
```

Out[293…
```
0                .@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Au
stin, it was dead!  I need to upgrade. Plugin stations at #SXSW.
1    @jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely
appreciate for its design. Also, they're giving free Ts at #SXSW
2                                                          @swonderlin Can
not wait for #iPad 2 also. They should sale them down at #SXSW.
3                                                       @sxsw I hope this
year's festival isn't as crashy as this year's iPhone app. #sxsw
4            @sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim
O'Reilly (tech books/conferences) &amp;amp; Matt Mullenweg (Wordpress)
Name: cleaned, dtype: object
```

In [294…
```
# check for duplicates
df.duplicated(subset=['tweet_text'], keep='first').sum()
```

Out[294…
```
27
```

In [295…
```
# Take a look at duplicate rows
dupes = df.duplicated(subset=['tweet_text'], keep=False)
df.loc[dupes.loc[dupes==True].index].sort_values(by='tweet_text')
```

Out[295…

	tweet_text	product	emotion	cleaned
7	#SXSW is just starting, #CTIA is around the corner and #googleio is only a hop skip and a jump from there, good time to be an #android fan	Android	Positive emotion	#SXSW is just starting, #CTIA is around the corner and #googleio is only a hop skip and a jump from there, good time to be an #android fan

	tweet_text	product	emotion	cleaned
3962	#SXSW is just starting, #CTIA is around the corner and #googleio is only a hop skip and a jump from there, good time to be an #android fan	Android	Positive emotion	#SXSW is just starting, #CTIA is around the corner and #googleio is only a hop skip and a jump from there, good time to be an #android fan
466	Before It Even Begins, Apple Wins #SXSW {link}	Apple	Positive emotion	Before It Even Begins, Apple Wins #SXSW {link}
468	Before It Even Begins, Apple Wins #SXSW {link}	Apple	Positive emotion	Before It Even Begins, Apple Wins #SXSW {link}
9	Counting down the days to #sxsw plus strong Canadian dollar means stock up on Apple gear	Apple	Positive emotion	Counting down the days to #sxsw plus strong Canadian dollar means stock up on Apple gear
2559	Counting down the days to #sxsw plus strong Canadian dollar means stock up on Apple gear	Apple	Positive emotion	Counting down the days to #sxsw plus strong Canadian dollar means stock up on Apple gear
774	Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw	NaN	No emotion toward brand or product	Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw
776	Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw	NaN	No emotion toward brand or product	Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw
17	I just noticed DST is coming this weekend. How many iPhone users will be an hour late at SXSW come Sunday morning? #SXSW #iPhone	iPhone	Negative emotion	I just noticed DST is coming this weekend. How many iPhone users will be an hour late at SXSW come Sunday morning? #SXSW #iPhone
8483	I just noticed DST is coming this weekend. How many iPhone users will be an hour late at SXSW come Sunday morning? #SXSW #iPhone	iPhone	Negative emotion	I just noticed DST is coming this weekend. How many iPhone users will be an hour late at SXSW come Sunday morning? #SXSW #iPhone
2230	Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw	NaN	No emotion toward brand or product	Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw
2232	Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw	NaN	No emotion toward brand or product	Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw
8747	Need to buy an iPad2 while I'm in Austin at #sxsw. Not sure if I'll need to Q up at an Austin Apple store?	iPad	Positive emotion	Need to buy an iPad2 while I'm in Austin at #sxsw. Not sure if I'll need to Q up at an Austin Apple store?
20	Need to buy an iPad2 while I'm in Austin at #sxsw. Not sure if I'll need to Q up at an Austin Apple store?	iPad	Positive emotion	Need to buy an iPad2 while I'm in Austin at #sxsw. Not sure if I'll need to Q up at an Austin Apple store?

	tweet_text	product	emotion	cleaned
4897	Oh. My. God. The #SXSW app for iPad is pure, unadulterated awesome. It's easier to browse events on iPad than on the website!!!	iPad or iPhone App	Positive emotion	Oh. My. God. The #SXSW app for iPad is pure, unadulterated awesome. It's easier to browse events on iPad than on the website!!!
21	Oh. My. God. The #SXSW app for iPad is pure, unadulterated awesome. It's easier to browse events on iPad than on the website!!!	iPad or iPhone App	Positive emotion	Oh. My. God. The #SXSW app for iPad is pure, unadulterated awesome. It's easier to browse events on iPad than on the website!!!
5884	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #SXSW	NaN	No emotion toward brand or product	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #SXSW
5882	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #SXSW	NaN	No emotion toward brand or product	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #SXSW
5880	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #SXSW	NaN	No emotion toward brand or product	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #SXSW
5883	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw	NaN	No emotion toward brand or product	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw
5879	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw	NaN	No emotion toward brand or product	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw
5881	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw	NaN	No emotion toward brand or product	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw
5885	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw	NaN	No emotion toward brand or product	RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw
6295	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #SXSW	NaN	No emotion toward brand or product	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #SXSW
6293	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #SXSW	Google	Positive emotion	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #SXSW

	tweet_text	product	emotion	cleaned
6297	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #SXSW	NaN	No emotion toward brand or product	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #SXSW
6299	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #SXSW	NaN	No emotion toward brand or product	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #SXSW
6296	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw	Google	Positive emotion	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw
6294	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw	NaN	No emotion toward brand or product	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw
6292	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw	Google	Positive emotion	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw
6298	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw	Google	Positive emotion	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw
6300	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw	NaN	No emotion toward brand or product	RT @mention Marissa Mayer: Google Will Connect the Digital & Physical Worlds Through Mobile - {link} #sxsw
6544	RT @mention RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw	NaN	No emotion toward brand or product	RT @mention RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw
6546	RT @mention RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw	NaN	No emotion toward brand or product	RT @mention RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw
6576	RT @mention RT @mention It's not a rumor: Apple is opening up a temporary store in downtown Austin for #SXSW and the iPad 2 launch {link}	NaN	No emotion toward brand or product	RT @mention RT @mention It's not a rumor: Apple is opening up a temporary store in downtown Austin for #SXSW and the iPad 2 launch {link}
6574	RT @mention RT @mention It's not a rumor: Apple is opening up a temporary store in downtown Austin for #SXSW and the iPad 2 launch {link}	Apple	Positive emotion	RT @mention RT @mention It's not a rumor: Apple is opening up a temporary store in downtown Austin for #SXSW and the iPad 2 launch {link}

	tweet_text	product	emotion	cleaned
5338	RT @mention ‰÷¼ GO BEYOND BORDERS! ‰÷_ {link} ‰ã_ #edchat #musedchat #sxsw #sxswi #classical #newTwitter	NaN	No emotion toward brand or product	RT @mention ‰÷¼ GO BEYOND BORDERS! ‰÷_ {link} ‰ã_ #edchat #musedchat #sxsw #sxswi #classical #newTwitter
5336	RT @mention ‰÷¼ GO BEYOND BORDERS! ‰÷_ {link} ‰ã_ #edchat #musedchat #sxsw #sxswi #classical #newTwitter	NaN	No emotion toward brand or product	RT @mention ‰÷¼ GO BEYOND BORDERS! ‰÷_ {link} ‰ã_ #edchat #musedchat #sxsw #sxswi #classical #newTwitter
5339	RT @mention ‰÷¼ Happy Woman's Day! Make love, not fuss! ‰÷_ {link} ‰ã_ #edchat #musedchat #sxsw #sxswi #classical #newTwitter	NaN	No emotion toward brand or product	RT @mention ‰÷¼ Happy Woman's Day! Make love, not fuss! ‰÷_ {link} ‰ã_ #edchat #musedchat #sxsw #sxswi #classical #newTwitter
5341	RT @mention ‰÷¼ Happy Woman's Day! Make love, not fuss! ‰÷_ {link} ‰ã_ #edchat #musedchat #sxsw #sxswi #classical #newTwitter	NaN	No emotion toward brand or product	RT @mention ‰÷¼ Happy Woman's Day! Make love, not fuss! ‰÷_ {link} ‰ã_ #edchat #musedchat #sxsw #sxswi #classical #newTwitter
3950	Really enjoying the changes in Gowalla 3.0 for Android! Looking forward to seeing what else they & Foursquare have up their sleeves at #SXSW	Android App	Positive emotion	Really enjoying the changes in Gowalla 3.0 for Android! Looking forward to seeing what else they & Foursquare have up their sleeves at #SXSW
24	Really enjoying the changes in Gowalla 3.0 for Android! Looking forward to seeing what else they & Foursquare have up their sleeves at #SXSW	Android App	Positive emotion	Really enjoying the changes in Gowalla 3.0 for Android! Looking forward to seeing what else they & Foursquare have up their sleeves at #SXSW
3814	Win free iPad 2 from webdoc.com #sxsw RT	iPad	Positive emotion	Win free iPad 2 from webdoc.com #sxsw RT
3812	Win free iPad 2 from webdoc.com #sxsw RT	NaN	No emotion toward brand or product	Win free iPad 2 from webdoc.com #sxsw RT
3813	Win free ipad 2 from webdoc.com #sxsw RT	iPad	Positive emotion	Win free ipad 2 from webdoc.com #sxsw RT
3811	Win free ipad 2 from webdoc.com #sxsw RT	NaN	No emotion toward brand or product	Win free ipad 2 from webdoc.com #sxsw RT

Definitely seeing some inconsistencies in coding the exact same tweets. Keeping this in mind, but going to remove the duplicates.

In [296…

```
# drop duplicates
df.drop_duplicates(subset=['tweet_text'], keep='first', inplace=True)
# check for duplicates
df.duplicated(subset=['tweet_text'], keep='first').sum()
```

Out[296… 0

In [297…
```
df.reset_index(drop=True, inplace=True)
```

Clean non-ASCII, HTML charcodes, and links

In [298…
```
# check for literal (unescaped) open or closing HTML tags
df[df['tweet_text'].str.contains("[<>]")]
```

Out[298…

	tweet_text	product	emotion	cleaned

Since there are no "<>" characters in any of the tweets, I will not add anything to remove HTML tags from the text when I clean it.

In [299…
```
# check for links or URLs
df[df['tweet_text'].str.contains("http[^ ]+|www\.[^ ]+")]
```

Out[299…

	tweet_text	product	emotion	
5	@teachntech00 New iPad Apps For #SpeechTherapy And Communication Are Showcased At The #SXSW Conference http://ht.ly/49n4M #iear #edchat #asd	NaN	No emotion toward brand or product	@teachntech00 New iP #SpeechTherapy And Commu Showcased At The #SXSW http://ht.ly/49n4M #iear #e
7	Beautifully smart and simple idea RT @madebymany @thenextweb wrote about our #hollergram iPad app for #sxsw! http://bit.ly/ieaVOB	iPad or iPhone App	Positive emotion	Beautifully smart and sim @madebymany @thenextweb our #hollergram iPad ap http://b
10	Find & Start Impromptu Parties at #SXSW With @HurricaneParty http://bit.ly/gVLrIn I can't wait til the Android app comes out.	Android App	Positive emotion	Find & Start Impromp #SXSW With @Hu http://bit.ly/gVLrIn I can't wait til app
11	Foursquare ups the game, just in time for #SXSW http://j.mp/grN7pK) - Still prefer @Gowalla by far, best looking Android app to date.	Android App	Positive emotion	Foursquare ups the game, jus #SXSW http://j.mp/grN7pK) @Gowalla by far, best looking An
12	Gotta love this #SXSW Google Calendar featuring top parties/ show cases to check out. RT @hamsandwich via @ischafer =>http://bit.ly/aXZwxB	Other Google product or service	Positive emotion	Gotta love this #SXSW Goog featuring top parties/ show cas out. RT @hamsandwich vi =>http://bi
13	Great #sxsw ipad app from @madebymany: http://tinyurl.com/4nqv92l	iPad or iPhone App	Positive emotion	Great #sxsw ipad app from @m http://tinyurl.c
14	haha, awesomely rad iPad app by @madebymany http://bit.ly/hTdFim #hollergram #sxsw	iPad or iPhone App	Positive emotion	haha, awesomely rad @madebymany http://b #holler
15	Holler Gram for iPad on the iTunes App Store - http://t.co/kfN3f5Q (via @marc_is_ken) #sxsw	NaN	No emotion toward brand or product	Holler Gram for iPad on the iTune - http://t.co/kfN3f5Q (via @m

	tweet_text	product	emotion	
18	Must have #SXSW app! RT @malbonster: Lovely review from Forbes for our SXSW iPad app Holler Gram - http://t.co/g4GZypV	iPad or iPhone App	Positive emotion	Must have #SXSW app! RT @ Lovely review from Forbes for our app Holler Gram - http://t.c
22	Photo: Just installed the #SXSW iPhone app, which is really nice! http://tumblr.com/x6t1pi6av7	iPad or iPhone App	Positive emotion	Photo: Just installed the #SXSW which is http://tumblr.com
25	RT haha, awesomely rad iPad app by @madebymany http://bit.ly/hTdFim #hollergram #sxsw (via @michaelpiliero)	iPad or iPhone App	Positive emotion	RT haha, awesomely rad @madebymany http://b #hollergram #sxsw (via @mic
27	The new #4sq3 looks like it is going to rock. Update for iPhone and Android should push tonight http://bit.ly/etsbZk #SXSW #KeepAustinWeird	iPad or iPhone App	Positive emotion	The new #4sq3 looks like it is go Update for iPhone and Android s tonight http://bit.ly/ets #Keep
29	Very smart from @madebymany #hollergram iPad app for #sxsw! http://t.co/A3xvWc6 (may leave my vuvuzela at home now)	iPad or iPhone App	Positive emotion	Very smart from @madebymany iPad app for #sxsw! http://t.co/A3x leave my vuvuzela at
30	You must have this app for your iPad if you are going to #SXSW http://itunes.apple.com/us/app/holler-gram/id420666439?mt=8 #hollergram	iPad or iPhone App	Positive emotion	You must have this app for you are goin http://itunes.apple.com/us gram/id420666439?mt=8
271	Surprise! Apple has opened a pop-up store in Austin so that the nerds in town for #SXSW can get their new iPads. http:/ {link}	NaN	No emotion toward brand or product	Surprise! Apple has opened a pop Austin so that the nerds in towr can get their new iPads.
292	A special Apple store: opening at 6th and Congress for SXSW & ipad 2 launch. www.apple.com/retail/thedomain/ #Apple #iPad2 #sxsw #fb	NaN	No emotion toward brand or product	A special Apple store: openin Congress for SXSW & ipa www.apple.com/retail/thedom #iPad
1130	Check out the @mention Route {link} ; RSVP here -> https://www.facebook.com/event.php?eid=141164002609303 #sxswi #sxsw	NaN	No emotion toward brand or product	Check out the @mention Route { https://www.facebook.com eid=141164002609303 #s
1136	Pics from the #apple #ipad2 line at #SXSW #fb {link} {link} http://t.co/26SVO3m	NaN	No emotion toward brand or product	Pics from the #apple #ipad2 lin #fb {link} {link} http://t.c
1238	Apple set to open popup shop in core of SXSW action {link} #sxsw #app... {link} http://bit.ly/bvHD5s	NaN	No emotion toward brand or product	Apple set to open popup sho SXSW action {link} #sxsw # http://bi
1755	I'll pay $681.00 for a New, Unopened iPad 2 16GB with 3G for Verizon - Black. Visit www.Zaarly.com to claim the cash. #willpay #sxsw	NaN	No emotion toward brand or product	I'll pay $681.00 for a New, Unop 16GB with 3G for Verizon - www.Zaarly.com to claim the ca

	tweet_text	product	emotion	
2731	Front Gate Tickets Present The Morning After Party 3/18 https://sites.google.com/site/frontgatesxsw11/ #SXSW Music	NaN	No emotion toward brand or product	Front Gate Tickets Present The M https://sites.google.com/site/front #S
3506	At #tweethouse watching #ipad dj @mention rock out at #sxsw! Check out www.rana.co! #dotco	NaN	No emotion toward brand or product	At #tweethouse watching #ipad d rock out at #sxsw! Check out w
4952	Off to Google party with @mention then @mention @mention @mention at www.getdown.com #msusxsw #sxsw Tweet me if you're there! (@jeremie)	NaN	No emotion toward brand or product	Off to Google party with @m @mention @mention @ www.getdown.com #msusxsw # me if you're there!
4992	RT The coolest iPhone 4 & iPad cases at #Sxsw, check them out here at #fastcompanygrill #zazzlesxsw #sxswi http... {link}	NaN	No emotion toward brand or product	RT The coolest iPhone 4 & iP #Sxsw, check them #fastcompanygrill #zazzlesxsw #
5120	RT @mention @mention hey Rudy, are Belgiums and Dutch taking over #sxsw? We launched www.skylines.net and {link} today	NaN	No emotion toward brand or product	RT @mention @mention he Belgiums and Dutch taking over launched www.skylines.net and
5358	RT @mention A special Apple store: opening at 6th and Congress for SXSW & ipad 2 launch. www.apple.com/retail/thedomain/ #Apple #iPad2 #sxsw #fb	NaN	No emotion toward brand or product	RT @mention A special Apple sto at 6th and Congress for SXSW & launch. www.apple.com/retail #Apple #iPad2
5476	RT @mention Are you at #sxsw? Check out #Tokii in the the Maple Leaf Digital Lounge {link} www.tokii.com	NaN	No emotion toward brand or product	RT @mention Are you at #sxsw #Tokii in the the Maple Leaf Dig {link} ww
5509	RT @mention At #tweethouse watching #ipad dj @mention rock out at #sxsw! Check out www.rana.co! #dotco	iPad	Positive emotion	RT @mention At #tweethouse wat dj @mention rock out at #sxsw www.rana
5596	RT @mention Check out the @mention Route {link} ; RSVP here -> https://www.facebook.com/event.php?eid=141164002609303 #sxswi #sxsw	NaN	No emotion toward brand or product	RT @mention Check out the @me {link} ; RSVI https://www.facebook.com eid=141164002609303 #s
5717	RT @mention Follow our #SXSW coverage at {link} on mobile at {link} or with our iPhone app http://bit.ly/guardianapp	NaN	No emotion toward brand or product	RT @mention Follow our #SXSW {link} on mobile at {link} or with app http://bit.ly/g
5741	RT @mention Front Gate Tickets Present The Morning After Party 3/18 https://sites.google.com/site/frontgatesxsw11/ #SXSW Music	NaN	No emotion toward brand or product	RT @mention Front Gate Tickets Morning Afte https://sites.google.com/site/front #S

	tweet_text	product	emotion	
6454	RT @mention Retrollect is now also in Android Market! #SxSW https://market.android.com/details?id=com.borderstylo.retrollect	Android App	Positive emotion	RT @mention Retrollect is Android Mar https://market.android.c id=com.borderstyl
6853	RT @mention We can't wait to give an iPad to someone at #sxsw. Want in? Just head to www.pep.jobs/upc to enter. (must be present to win	iPad	Positive emotion	RT @mention We can't wait to giv someone at #sxsw. Want in? J www.pep.jobs/upc to enter. (must
7324	Saw a company today ready to launch, sounds a lot like Google Circles, but with actual personal privacy www.mycube.com #sxsw	Other Google product or service	No emotion toward brand or product	Saw a company today read sounds a lot like Google Circl actual personal privacy www.m
7495	Follow our #SXSW coverage at {link} on mobile at {link} or with our iPhone app http://bit.ly/guardianapp	iPad or iPhone App	No emotion toward brand or product	Follow our #SXSW coverage mobile at {link} or with our http://bit.ly/g
7695	miami horror, tacos and bloody marys. i'm there. https://sites.google.com/site/frontgatesxsw11/ #sxsw	NaN	No emotion toward brand or product	miami horror, tacos and blood https://sites.google.com/site/front
7776	Per this rumor, Google may preview its big social strategy at an '80s-themed costume party at #SXSW. Yep. http:/... {link}	NaN	No emotion toward brand or product	Per this rumor, Google may pre social strategy at an '80s-them party at #SXSW. Yep. h
8021	Adloopz: Social Media Advertising done the easy way. Visit www.adloopz.com #SXSW #sxswi #ipad	NaN	No emotion toward brand or product	Adloopz: Social Media Advertisi easy way. Visit www.adloopz.c #
8115	@mention Sweet 8-bit pic. Are you an artist? www.zaggle.org is having an #art contest for our #iPhone app Check it out We'll be at #SXSW	NaN	No emotion toward brand or product	@mention Sweet 8-bit pic. Are yo www.zaggle.org is having an #art our #iPhone app Check it ou
8175	Are you at #sxsw? Check out #Tokii in the the Maple Leaf Digital Lounge {link} www.tokii.com	NaN	No emotion toward brand or product	Are you at #sxsw? Check out #Tol Maple Leaf Digital L ww
8224	Retrollect is now also in Android Market! #SxSW https://market.android.com/details?id=com.borderstylo.retrollect	NaN	Positive emotion	Retrollect is now also in Andr #SxSW https://market.android.c id=com.borderstyl
8821	We can't wait to give an iPad to someone at #sxsw. Want in? Just head to www.pep.jobs/upc to enter. (must be present to win)	iPad	No emotion toward brand or product	We can't wait to give an iPad to #sxsw. Want in? J www.pep.jobs/upc to enter. (must

I definitely do have URLs, which I will remove.

In [300…

```
# check for non-ASCII characters
# regex from https://stackoverflow.com/questions/2124010/grep-regex-to-match-non

df[df['tweet_text'].str.contains('[^\x00-\x7F]+')]
```

Out[300…

	tweet_text	product	emotion	
37	@mention - False Alarm: Google Circles Not Coming Now‰ÛÒand Probably Not Ever? - {link} #Google #Circles #Social #SXSW	Google	Negative emotion	@mention - False Alarm: (Coming Now‰ÛÒand Probabl #Google #Circl
40	HootSuite - HootSuite Mobile for #SXSW ~ Updates for iPhone, BlackBerry & Android: Whether you‰Ûªre getting friend... {link}	NaN	No emotion toward brand or product	HootSuite - HootSuite M Updates for iPhone, BlackBer Whether you‰Ûªre ge
41	Hey #SXSW - How long do you think it takes us to make an iPhone case? answer @mention using #zazzlesxsw and we‰Ûªll make you one!	NaN	No emotion toward brand or product	Hey #SXSW - How long do y to make an iPhone case? using #zazzlesxsw and we‰
44	#IPad2 's ‰Û÷#SmartCover‰Ûª Opens to Instant Access - I should have waited to get one! - {link} #apple #SXSW	iPad or iPhone App	Positive emotion	#IPad2 's ‰Û÷#SmartCover‰Û Access - I should have waited
45	Hand-Held ‰Û÷Hobo‰Ûª: Drafthouse launches ‰Û÷Hobo With a Shotgun‰Ûª iPhone app #SXSW {link}	NaN	Positive emotion	Hand-Held ‰Û÷Hobo‰Ûª: Dr ‰Û÷Hobo With a Shotgun‰Ûª iI
...	...	...	...	
8897	umm that would be @mention ‰ÛÏ@mention I keep winning shit! Thanks @mention for the killer iPad case. #sxsw‰Û	Other Apple product or service	Positive emotion	umm that would be @ment keep winning shit! Thanks killer iPa
8917	FestivalExplorer iPhone App Finally Solves SXSW {link} #music #musica #musiek #musique #musik #app #sxsw # Ù_¾¬â # Ù_¾´_ #“ Î“¥É	iPad or iPhone App	Positive emotion	FestivalExplorer iPhone App Fi {link} #music #musica # #musik #app #sxsw # Ù_¾¬â
8935	Group #Texting War Heats Up: Fast Society Launches New Android App, Updates iPhone App: #SXSW‰Û_ {link}	Android App	Positive emotion	Group #Texting War Heat Launches New Android Ap App
8954	In case my fairy god mother = reading mail; my ÌÙ€±G wish this week is 2 go 2 #sxsw ‰Ï for the #Android ‰ÌÎ Dev ‰Ïà Meetup. @mention Hilton, Sat....	NaN	No emotion toward brand or product	In case my fairy god mother ÌÙ€±G wish this week is 2 go #Android ‰ÌÎ Dev ‰Ïà Meetup
9064	Œ¡ ÏàŠü_‹ Ê‹ Î‹ Ò‹ £‹ Á‹ââ‹ _‹ £‹ ‹â_‹ÛâRT @mention Google Tests ‰ÛÏCheck-in Offers‰Û At #SXSW {link}	NaN	No emotion toward brand or product	Œ¡ ÏàŠü_‹ Ê‹ Î‹ Ò‹ £‹ Á‹ââ‹ @mention Google Tests ‰ÛÏC

483 rows × 4 columns

In [301…

```
# How would the encode/decode trick work? would it give me better text than
# simply removing the non-ASCII chars?
```

```
doc = df.at[38, 'tweet_text']
doc = doc.encode('ascii', 'ignore').decode()
doc
```

Out[301… `'VatorNews - Google And Apple Force Print Media to Evolve? {link} #sxsw'`

The problem I have with the non-ASCII characters is that they often seem to represent apostrophes and quotation marks. I can't find an encoding that will show them properly in python, or in a SublimeText when I open the raw file.

For now I'm going to replace the non-ASCII characters with spaces because I think replacing them with nothing will lead to weird words.

In [302…
```
def clean_docs(doc):
    """
    Performs a few basic cleaning steps:
    - Unescapes any escaped HTML (i.e. `&amp;` for ampersand)
    - Replaces URLs with the `{link}` placeholder text
    - Replaces non-ASCII characters with spaces
    - Replaces any control characters with spaces
    - Finally, cleans up any instances of multiple spaces that might have
    been created during this process, replacing with a single space

    Takes in a document at a time, and returns the cleaned document text as
    a string.
    """
    # unescape HTML characters
    doc = html.unescape(doc)

    # remove URLs and links, replacing them with existing placeholder
    urls = re.findall("http[^ ]+|www\.[^ ]+", doc)
    for url in urls:
        doc = str.replace(doc, url, '{link}')

    # replace non-ASCII characters with space
    doc = re.sub(r"[^\x00-\x7F]+", ' ', doc)

    # replace ASCII control characters with space
    doc = re.sub(r"[\x00-\x1F]", ' ', doc)

    # remove multiple spaces, which will exist after all this replacing words
    doc = re.sub(r"[ ]{2,}", ' ', doc)

    return doc
```

In [303…
```
# map doc cleaning function onto 'cleaned' column
df['cleaned'] = df['cleaned'].map(lambda x: clean_docs(x))
df
```

Out[303…

	tweet_text	product	emotion	cleaned
0	.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.	iPhone	Negative emotion	.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.

	tweet_text	product	emotion	cleaned
1	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW	iPad or iPhone App	Positive emotion	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	iPad	Positive emotion	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.
3	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	iPad or iPhone App	Negative emotion	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw
4	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	Google	Positive emotion	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)
...	...	...	...	...
9060	Ipad everywhere. #SXSW {link}	iPad	Positive emotion	Ipad everywhere. #SXSW {link}
9061	Wave, buzz... RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles	NaN	No emotion toward brand or product	Wave, buzz... RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
9062	Google's Zeiger, a physician never reported potential AE. Yet FDA relies on physicians. "We're operating w/out data." #sxsw #health2dev	NaN	No emotion toward brand or product	Google's Zeiger, a physician never reported potential AE. Yet FDA relies on physicians. "We're operating w/out data." #sxsw #health2dev
9063	Some Verizon iPhone customers complained their time fell back an hour this weekend. Of course they were the New Yorkers who attended #SXSW.	NaN	No emotion toward brand or product	Some Verizon iPhone customers complained their time fell back an hour this weekend. Of course they were the New Yorkers who attended #SXSW.
9064	¢Ï¡ ÏàŠü_‹ Ê‹ Î‹ Ò‹ £‹ Á‹ââ‹ _‹ £‹ ‹â_‹ÛâRT @mention Google Tests ‰ÛÏCheck-in Offers‰Û At #SXSW {link}	NaN	No emotion toward brand or product	_ _ _ RT @mention Google Tests Check-in Offers At #SXSW {link}

9065 rows × 4 columns

In [304…

```
# specifically check on a few rows
df.loc[[44, 8944, 8981]]
```

Out[304…

	tweet_text	product	emotion	cleaned
44	#IPad2 's ‰Û÷#SmartCover‰Ûª Opens to Instant Access - I should have waited to get one! - {link} #apple #SXSW	iPad or iPhone App	Positive emotion	#IPad2 's #SmartCover Opens to Instant Access - I should have waited to get one! - {link} #apple #SXSW
8944	Biomimicry as the basis of design and problem solving. Google studying flocking and swarming behavior to understand collab. Brilliant. #sxsw	Google	Positive emotion	Biomimicry as the basis of design and problem solving. Google studying flocking and swarming behavior to understand collab. Brilliant. #sxsw
8981	Very happy that Discovr has been named as one of the top ten must-have apps for iPad 2 :) I just have to fight my way thru the queue! #sxsw	iPad or iPhone App	Positive emotion	Very happy that Discovr has been named as one of the top ten must-have apps for iPad 2 :) I just have to fight my way thru the queue! #sxsw

In [305…

```
# check for links or URLs after cleaning
df[df['cleaned'].str.contains("http[^ ]+|www\.[^ ]+")]
```

Out[305…

	tweet_text	product	emotion	cleaned

In [306…

```
def get_pattern_hits(doc, pattern, out_type):
    """Takes in a regex pattern and a string of text, and checks for the
    presence of the pattern in the string.

    Returns a copy of the text string with the pattern replaced with spaces,
    and the matches. See `out_type` for available formats for the match output.

    *** Arguments

    doc: string. Text to be searched for the pattern.

    pattern: string, regex pattern. Pattern to be searched for in the string.

    out_type: string. Indicate how / whether the matches on the pattern
    should be returned.

        If `out_type` = `list`, each match on the pattern is saved into a list
        and the list is returned.

        If `out_type` = `string`, each match on the pattern is saved into a
        string separated by spaces and that string is returned.

        If `out_type` = `boolean`, the output is just
        True if there was at least one match, and false if there were none.

        If `out_type` = `none`, the pattern matches are simply removed from the
        text string and not logged in any way.
    """

    # determine the variable type for recording hits
    if out_type=='list':
        hits = []
    elif out_type=='string':
        hits = ""
    elif out_type=='bool':
```

```
        hits = False
    elif out_type=='none':
        hits = None

    # search for regex pattern in doc
    pattern_hits = re.findall(pattern, doc)

    if len(pattern_hits) > 0:
        # replace the hits in the original doc string
        # need to use the re version otherwise substrings won't be replaced
        # correctly!
        doc = re.sub(pattern, ' ', doc)

        # replace multiple spaces with a single space
        doc = re.sub(r"(\s{2,})", ' ', doc)

        # Update appropriate hits variable
        for hit in pattern_hits:
            if out_type=='list':
                hits.append(hit)
            elif out_type=='string':
                hits = hits + ' ' + hit
            elif out_type=='bool':
                hits = True

        if out_type=='list':
            hits = list(set(hits))

    return doc, hits
```

In [307…

```
def pattern_match_in_df(df, doc_col, hit_col, pattern, out_type='list',
                        replace=True):
    """Loops through values in a particular dataframe columns, and searches
    for regex pattern matches.

    Returns the same dataframe, updated with new `hit_col`, and replaced
    `doc_col`, depending on arguments passed.

    *** Arguments

    df: Dataframe containing the `doc_col` to be searched.

    doc_col: String. Name of the dataframe column containing the text to be
    searched for the pattern.

    hit_col: String. Name of the column which should be added to the dataframe
    when it's returned, containing the pattern hit information.

    pattern: string, regex pattern. Pattern to be searched for in the string.

    out_type: string. See `get_pattern_hits` documentation for details.

    replace: Boolean, default True.

    Use `replace=True` to have the matches be replaced with spaces, and the
    original `doc_col` replaced with a new one with matches removed.
    If `replace=False`, the `doc_col` will not be updated, and matches will
    only be logged in the new `hit_col`.

    """
```

```
    updates = []

    # loop through each row in the dataframe to process its record
    for i in df.index:
        new_doc, hits = get_pattern_hits(df.at[i, doc_col], pattern, out_type)
        updates.append([new_doc, hits])

    # create a dataframe out of the updated info
    df_new = pd.DataFrame(updates, columns=[doc_col, hit_col])

    # if we were told not to output the hits, drop the hits column
    if out_type=='none':
        df_new.drop(columns=[hit_col], inplace=True)

    # if we were told not to replace the hits in the original text column,
    # drop the new doc column
    if not replace:
        df_new.drop(columns=[doc_col], inplace=True)

    if len(df_new.columns) > 0:
        df = df.join(df_new, lsuffix='_old', how='inner')
        if replace:
            df.drop(columns=[f"{doc_col}_old"], inplace=True)
    else:
        print("Dataframe was returned as-is based on arguments passed.")

    return df
```

Remove words that are all numbers

In [308…

```
# remove numbers
df = pattern_match_in_df(df, doc_col='cleaned', hit_col='_',
                pattern=r"(?:^|\s)([.:$%]*[0-9]+[.:$%]*[0-9]*)\b",
                out_type='none', replace=True)
df[['tweet_text', 'cleaned']]
```

Out[308…

	tweet_text	cleaned
0	.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.	.@wesley83 I have a 3G iPhone. After hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.
1	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	@swonderlin Can not wait for #iPad also. They should sale them down at #SXSW.
3	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw
4	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)
…	…	…
9060	Ipad everywhere. #SXSW {link}	Ipad everywhere. #SXSW {link}

	tweet_text	cleaned
9061	Wave, buzz... RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles	Wave, buzz... RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
9062	Google's Zeiger, a physician never reported potential AE. Yet FDA relies on physicians. "We're operating w/out data." #sxsw #health2dev	Google's Zeiger, a physician never reported potential AE. Yet FDA relies on physicians. "We're operating w/out data." #sxsw #health2dev
9063	Some Verizon iPhone customers complained their time fell back an hour this weekend. Of course they were the New Yorkers who attended #SXSW.	Some Verizon iPhone customers complained their time fell back an hour this weekend. Of course they were the New Yorkers who attended #SXSW.
9064	¢Ï¡ ÏàŠü_‹ Ê‹ Î‹ Ò‹ £‹ Á‹ââ‹ _‹ £‹ ‹â_‹ÛâRT @mention Google Tests ‰ÛÏCheck-in Offers‰Û At #SXSW {link}	_ _ _ RT @mention Google Tests Check-in Offers At #SXSW {link}

9065 rows × 2 columns

In [309…
```
# I don't see many numbers in the head/tail cells, so let's find some
df.loc[df['tweet_text'].str.contains(r"(?:^|\s)([.:$%]*[0-9]+[.:$%]*[0-9]*)\s"),
       ['tweet_text', 'cleaned']]
```

```
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/pandas/core/strings.py:2001: UserWarning: This pattern has match groups. To actually get the groups, use str.extract.
  return func(self, *args, **kwargs)
```

Out[309…

	tweet_text	cleaned
0	.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.	.@wesley83 I have a 3G iPhone. After hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	@swonderlin Can not wait for #iPad also. They should sale them down at #SXSW.
23	Really enjoying the changes in Gowalla 3.0 for Android! Looking forward to seeing what else they & Foursquare have up their sleeves at #SXSW	Really enjoying the changes in Gowalla for Android! Looking forward to seeing what else they & Foursquare have up their sleeves at #SXSW
28	They were right, the @gowalla 3 app on #android is sweeeeet! Nice job by the team there. #sxsw	They were right, the @gowalla app on #android is sweeeeet! Nice job by the team there. #sxsw
42	Mashable! - The iPad 2 Takes Over SXSW [VIDEO] #ipad #sxsw #gadgets {link}	Mashable! - The iPad Takes Over SXSW [VIDEO] #ipad #sxsw #gadgets {link}
...	...	...
9024	@mention You could buy a new iPad 2 tmrw at the Apple pop-up store at #sxsw: {link}	@mention You could buy a new iPad tmrw at the Apple pop-up store at #sxsw: {link}
9026	Guys, if you ever plan on attending #SXSW, you need 4 things, skinny jeans, flannel shirt, beard and an iPad #imanoutcast...	Guys, if you ever plan on attending #SXSW, you need things, skinny jeans, flannel shirt, beard and an iPad #imanoutcast...
9035	@mention You should get the iPad 2 to save your back from lugging the laptop #SXSW #SXSWMyMistake	@mention You should get the iPad to save your back from lugging the laptop #SXSW #SXSWMyMistake

	tweet_text	cleaned
9044	@mention your iPhone 4 cases are Rad and Ready! Stop by tomorrow to get them! #Sxsw #zazzlesxsw #sxswi {link}	@mention your iPhone cases are Rad and Ready! Stop by tomorrow to get them! #Sxsw #zazzlesxsw #sxswi {link}
9048	At "Your Mom Has an iPad" session at #SXSW (@mention ACC - Ballroom B w/ 23 others) {link}	At "Your Mom Has an iPad" session at #SXSW (@mention ACC - Ballroom B w/ others) {link}

1663 rows × 2 columns

Log hashtags in a separate column

In [310…

```
df = pattern_match_in_df(df, doc_col='cleaned', hit_col='hashtags',
                         pattern=r"(?:^|\s)(#[a-zA-Z0-9_-]+)",
                         out_type='string', replace=False)
df[['tweet_text', 'cleaned', 'hashtags']]
```

Out[310…

	tweet_text	cleaned	hashtags
0	.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.	.@wesley83 I have a 3G iPhone. After hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.	#RISE_Austin #SXSW
1	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW	#SXSW
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	@swonderlin Can not wait for #iPad also. They should sale them down at #SXSW.	#iPad #SXSW
3	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	#sxsw
4	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	#SXSW
…	…	…	…
9060	Ipad everywhere. #SXSW {link}	Ipad everywhere. #SXSW {link}	#SXSW
9061	Wave, buzz... RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles	Wave, buzz... RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles	#sxsw #google #circles
9062	Google's Zeiger, a physician never reported potential AE. Yet FDA relies on physicians. "We're operating w/out data." #sxsw #health2dev	Google's Zeiger, a physician never reported potential AE. Yet FDA relies on physicians. "We're operating w/out data." #sxsw #health2dev	#sxsw #health2dev

	tweet_text	cleaned	hashtags
9063	Some Verizon iPhone customers complained their time fell back an hour this weekend. Of course they were the New Yorkers who attended #SXSW.	Some Verizon iPhone customers complained their time fell back an hour this weekend. Of course they were the New Yorkers who attended #SXSW.	#SXSW
9064	¢Ï¡ ÏàŠü_‹ Ê‹ Î‹ Ò‹ £‹ Á‹ââ‹ _‹ £‹ ‹â_‹ÛâRT @mention Google Tests ‰ÛÏCheck-in Offers‰Û At #SXSW {link}	_ _ _ RT @mention Google Tests Check-in Offers At #SXSW {link}	#SXSW

9065 rows × 3 columns

EXPLORE

Now that I've done some basic cleaning, I'd like to look at the characteristics of the corpus, such as:

- class balances
- distribution of tweet lengths
- most common words before and after removing stopwords and punctuation
- most common words in the different classes
- most common hashtags in the different classes

In [311…

```
def generate_wordcloud(docs, cmap, stopwords, min_font_size=14, n_grams=True,
                       title='Word cloud'):
    """Generate a wordcloud from a list of tokenized words. Words will be
    joined into a space-delimited string inside the function.
    """
    cloud = WordCloud(colormap=cmap, width=600, height=400,
                      prefer_horizontal=0.95, min_font_size=min_font_size,
                     collocations=n_grams)\
                      .generate_from_text(" ".join(docs))

    fig, ax = plt.subplots(figsize=(10, 6))

    # Display the generated image:
    ax.imshow(cloud)
    ax.set_axis_off()
    ax.set_title=(title)
    ax.margins(x=0, y=0);
```

In [312…

```
def generate_freqs_wordcloud(df, word_col, freq_col, cmap, min_font_size=14,
                             title='Word cloud'):
    """Generate a wordcloud from a dataframe where one column holds the words
    and another column holds the frequency or weight. The dictionary for the
    wordcloud is generated inside the function.
    """
    freq_dict = pd.Series(df[freq_col].values,index=df[word_col]).to_dict()

    cloud = WordCloud(colormap=cmap, width=600, height=400,
                      prefer_horizontal=0.95, min_font_size=min_font_size,
                     collocations=False).generate_from_frequencies(freq_dict)
```

```
    fig, ax = plt.subplots(figsize=(10, 6))

    # Display the generated image:
    ax.imshow(cloud)
    ax.set_axis_off()
    ax.set_title=(title)
    ax.margins(x=0, y=0);
```

In [313…

```
def plot_wordfreqs(df, word_col, freq_col, top_n, sub_title):

    with sns.plotting_context(context='talk'):
        fig, ax = plt.subplots(figsize=(8, top_n / 2.5))
        sns.barplot(data=df,
                    y=df[word_col][:top_n],
                    x=df[freq_col][:top_n], color='blue')
        ax.set_title(f"Top {top_n} Words\n{sub_title}")
```

In [314…

```
def tokenize_corpus_dict_tweet(df, target_vals, stop_list=None,
                               verbose=True, target_col='emotion',
                              doc_col='cleaned'):
    """ Tokenizes text and separates the tokens in a dictionary where the
    keys are target class labels.

    The column names are hard coded for this project.

    target_vals should be a list of the class labels in the `emotion` col.
    """
    tweettokenizer = TweetTokenizer(preserve_case=False, strip_handles=True)

    # generate corpus for each emotion
    corpus_per_target = {}

    for val in target_vals:

        if verbose:
            print(f"Starting target val: {val}")

        # get series of text docs per target_val
        docs = df.loc[df[target_col]==val, doc_col]

        # loop through docs and tokenize each one
        corpus = []

        i = 0
        for doc in docs:
            # tokenize using tweet tokenizer
            tokens = tweettokenizer.tokenize(doc)

            # remove stop words if needed
            if not stop_list == None:
                tokens = [token for token in tokens if token not in stop_list]

            # remove words if they're just spaces!
            tokens.remove(' ') if ' ' in tokens else None

            corpus.extend(tokens)
            i += 1

            if verbose and (i % 1000 == 0):
```

```
                print(f"Processed {i} docs out of {len(docs)}...")

        # add corpus to dict
        corpus_per_target[val] = corpus

    if verbose:
        print(f"Done!")
    return corpus_per_target
```

Generate initial stopwords and punctuation lists

In [315…

```
# starting with nltk's stopwords list
nltk_stopwords = stopwords.words('english')
nltk_stopwords.sort()
print(nltk_stopwords)
```

```
['a', 'about', 'above', 'after', 'again', 'against', 'ain', 'all', 'am', 'an',
'and', 'any', 'are', 'aren', "aren't", 'as', 'at', 'be', 'because', 'been', 'bef
ore', 'being', 'below', 'between', 'both', 'but', 'by', 'can', 'couldn', "could
n't", 'd', 'did', 'didn', "didn't", 'do', 'does', 'doesn', "doesn't", 'doing',
'don', "don't", 'down', 'during', 'each', 'few', 'for', 'from', 'further', 'ha
d', 'hadn', "hadn't", 'has', 'hasn', "hasn't", 'have', 'haven', "haven't", 'havi
ng', 'he', 'her', 'here', 'hers', 'herself', 'him', 'himself', 'his', 'how',
'i', 'if', 'in', 'into', 'is', 'isn', "isn't", 'it', "it's", 'its', 'itself', 'j
ust', 'll', 'm', 'ma', 'me', 'mightn', "mightn't", 'more', 'most', 'mustn', "mus
tn't", 'my', 'myself', 'needn', "needn't", 'no', 'nor', 'not', 'now', 'o', 'of',
'off', 'on', 'once', 'only', 'or', 'other', 'our', 'ours', 'ourselves', 'out',
'over', 'own', 're', 's', 'same', 'shan', "shan't", 'she', "she's", 'should', "s
hould've", 'shouldn', "shouldn't", 'so', 'some', 'such', 't', 'than', 'that', "t
hat'll", 'the', 'their', 'theirs', 'them', 'themselves', 'then', 'there', 'thes
e', 'they', 'this', 'those', 'through', 'to', 'too', 'under', 'until', 'up', 'v
e', 'very', 'was', 'wasn', "wasn't", 'we', 'were', 'weren', "weren't", 'what',
'when', 'where', 'which', 'while', 'who', 'whom', 'why', 'will', 'with', 'won',
"won't", 'wouldn', "wouldn't", 'y', 'you', "you'd", "you'll", "you're", "you've
e", 'your', 'yours', 'yourself', 'yourselves']
```

This has a lot of words that I think might be related to emotion towards a product, like negation (should / shouldn't, not), and words that can describe how or how much someone feels about something (so, under).

I'm going to create my own custom stopwords list that is much pared down.

In [316…

```
# my much-pared-down stopwords list for testing
custom_stopwords = ['a','an','and','am','are','as','at','be','by','for',
                    'from','if','in','is','into','it',"it's",'its','itself',
                    'of','on','or','than','that','the','to']
```

In [317…

```
# create full and custom punctuation list. Custom excludes ! and ?
punc = list(string.punctuation)
punc_custom = punc.copy()
punc_custom.remove('?')
punc_custom.remove('!')

print(punc)
print(punc_custom)
```

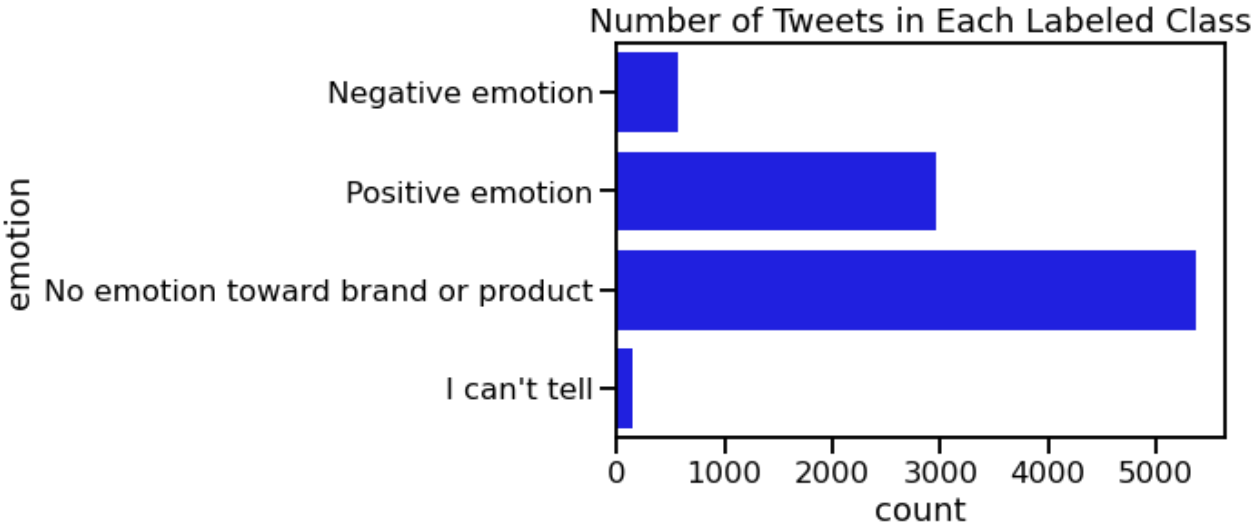
```
['!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', ':',
';', '<', '=', '>', '?', '@', '[', '\\', ']', '^', '_', '`', '{', '|', '}', '~']
['"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', ':', ';',
'<', '=', '>', '@', '[', '\\', ']', '^', '_', '`', '{', '|', '}', '~']
```

Explore Classes

In [318…
```
df['emotion'].value_counts()
```

Out[318…
```
No emotion toward brand or product    5372
Positive emotion                      2968
Negative emotion                       569
I can't tell                           156
Name: emotion, dtype: int64
```

In [319…
```
# Plot counts of each class
with sns.plotting_context(context='talk'):
    fig, ax = plt.subplots(figsize=(6, 4))
    sns.countplot(y=df['emotion'], orient='v', ax=ax, color='blue')
    ax.set_title('Number of Tweets in Each Labeled Class');
```



The vast majority of tweets are labeled 'No emotion towards brand or product'.

Only 570 are negative, and 2978 are positive. Even adding those together, there are 50% more tweets with no emotion towards a brand or product, compared to those that have some emotion towards a brand or product.

I'm assuming (although I can't know for sure) that this corpus represents tweets with #sxsw hashtags during the festival one year, and which also had keywords or hashtags related to Apple or Google products. I will see if I can confirm this during my EDA.

In [320…
```
# create a list of all the emotions
all_emotions = list(df['emotion'].value_counts().index)
all_emotions
```

Out[320…
```
['No emotion toward brand or product',
 'Positive emotion',
 'Negative emotion',
 "I can't tell"]
```

Tweet Lengths

In [321…
```
def count_words(doc):
    tokenizer_no_strip = TweetTokenizer(strip_handles=False)
    tokens = tokenizer_no_strip.tokenize(doc)
```

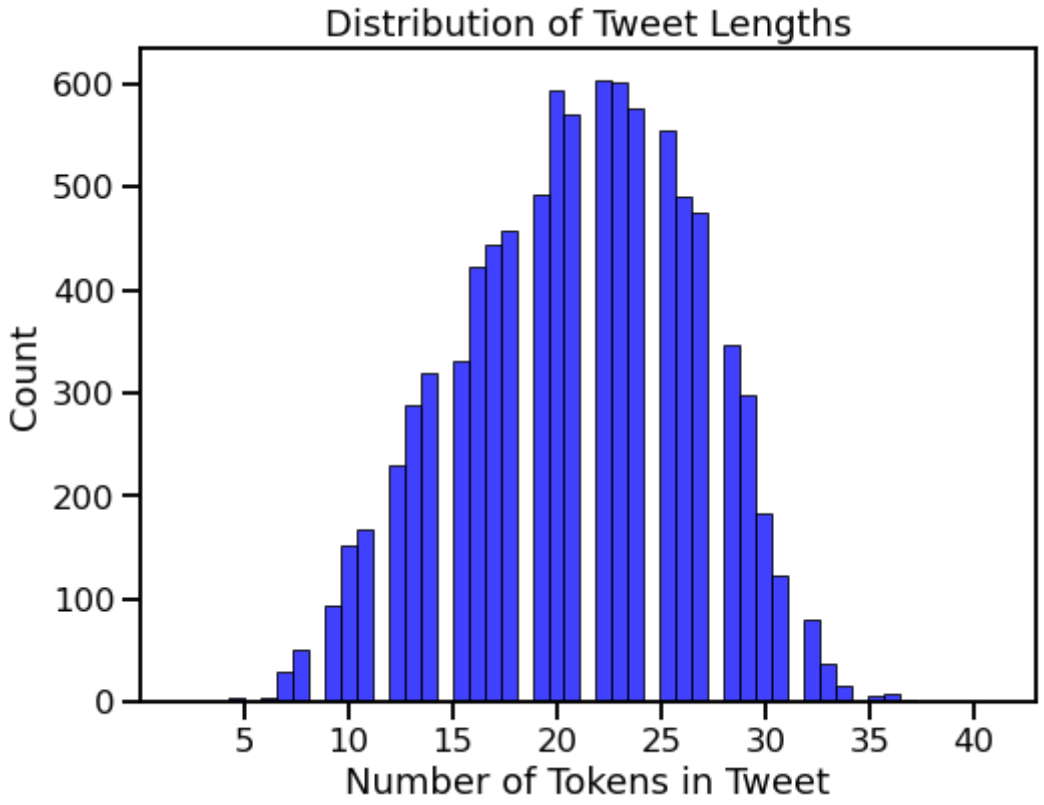
```
    return len(tokens)

df['raw_token_count'] = df['cleaned'].map(lambda x: count_words(x))
df.head()
```

Out[321…

	tweet_text	product	emotion	cleaned	hashtags	raw_token_count
0	.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.	iPhone	Negative emotion	.@wesley83 I have a 3G iPhone. After hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.	#RISE_Austin #SXSW	28
1	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW	iPad or iPhone App	Positive emotion	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW	#SXSW	26
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	iPad	Positive emotion	@swonderlin Can not wait for #iPad also. They should sale them down at #SXSW.	#iPad #SXSW	16
3	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	iPad or iPhone App	Negative emotion	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	#sxsw	16
4	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	Google	Positive emotion	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	#SXSW	27

In [322…

```
# visualize tweet length distribution
with sns.plotting_context(context='talk'):
    fig, ax = plt.subplots(figsize=(8, 6))
    sns.histplot(df['raw_token_count'], color='blue', ax=ax)
    ax.set_title('Distribution of Tweet Lengths')
    ax.set_xlabel('Number of Tokens in Tweet');
```



I see a pretty normal distribution here, where most tweets are 20-25 tokens long. This may be slightly different than the final words, since we haven't removed punctuation yet and right now it will be counted as a token in many cases.

Explore Entire Corpus

No stopwords or punctuation removed

In [323…
```
# Create a dictionary of tokens with class labels as keys
# removing punctuation only
token_dict = tokenize_corpus_dict_tweet(df, all_emotions,
                    stop_list=None, verbose=False)
token_dict['Positive emotion'][:5]
```

Out[323…
```
['know', 'about', '?', 'awesome', 'ipad']
```

In [324…
```
# create list of all word frequencies in the corpus
all_tokens = []
[all_tokens.extend(tokens) for tokens in token_dict.values()]

len(all_tokens)
```

Out[324…
```
182652
```

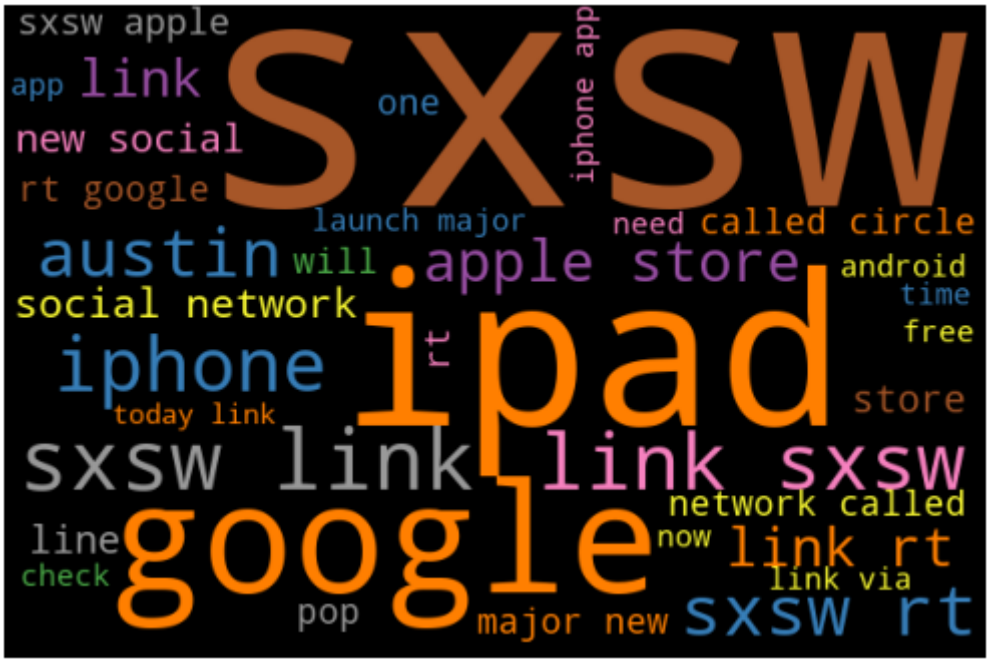
In [325…
```
# get frequency distribution for entire corpus
all_corpus = FreqDist(all_tokens)

corpus_freq_df = pd.DataFrame(all_corpus.most_common(100),
                              columns=['Word','Count'])
corpus_freq_df[:5]
```

Out[325…

	Word	Count
0	#sxsw	9069
1	.	5885
2	the	4420
3	link	4336
4	}	4321

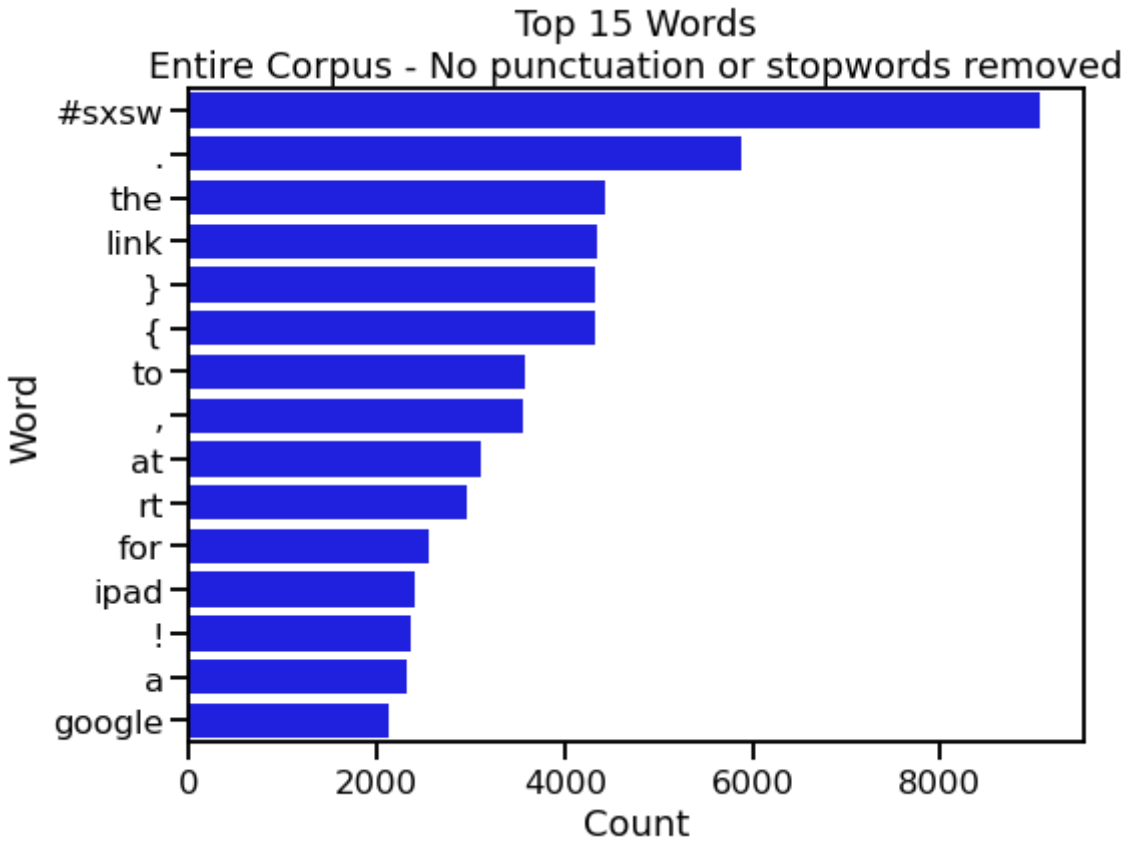
In [326…

```
# Wordcloud for most common words in whole corpus, no stopwords
# or punctuation removed
generate_wordcloud(docs=all_tokens, cmap="Set1",
                   stopwords=None, min_font_size=16,
                  title='Entire Corpus')
```



In [327…

```
# Plot top 15 words across the entire corpus, no punctuation removed
plot_wordfreqs(corpus_freq_df, 'Word', 'Count', 15,
                  "Entire Corpus - No punctuation or stopwords removed")
```



OK, the top words are mostly punctuation and stopwords.

I also see #sxsw, which isn't too surprising, since that's the event this corpus was focused on. I'm going to add #sxsw to the stopwords lists, and check out what we get after applying that and removing punctuation.

In [328…

```
# extend custom stopwords list to include information about the event,
# which will likely be in common across all tweets
nltk_stopwords.extend(['austin', 'sxsw', '#sxsw'])
custom_stopwords.extend(['austin', 'sxsw', '#sxsw'])
print(custom_stopwords)
```

```
['a', 'an', 'and', 'am', 'are', 'as', 'at', 'be', 'by', 'for', 'from', 'if', 'i
n', 'is', 'into', 'it', "it's", 'its', 'itself', 'of', 'on', 'or', 'than', 'tha
t', 'the', 'to', 'austin', 'sxsw', '#sxsw']
```

Minimal custom stopwords and punctuation

In [329…

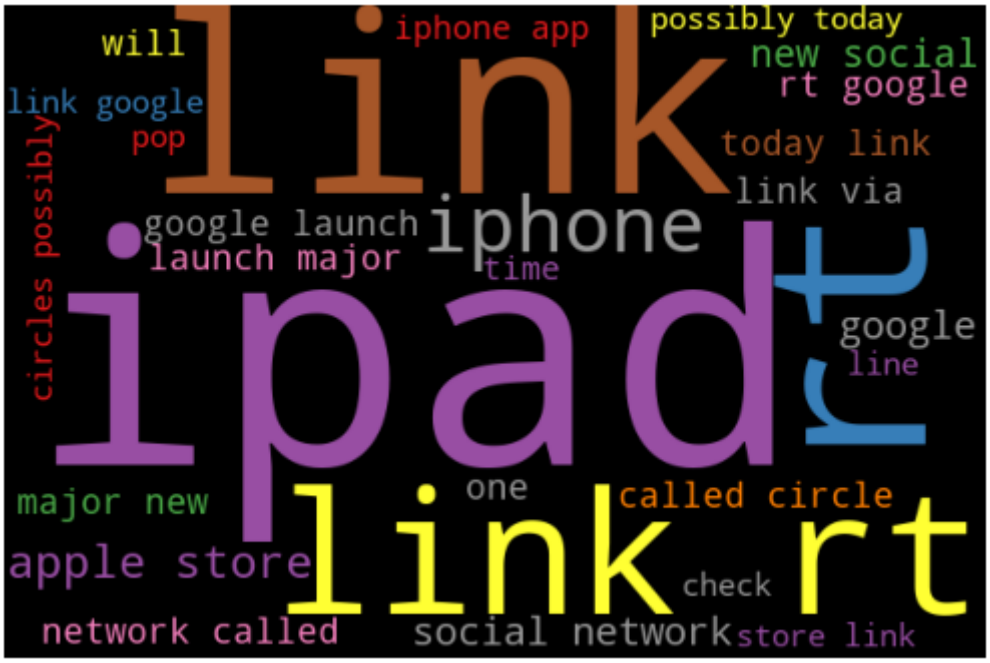
```
# make a new dict with stopwords and punctuation removed
token_dict = tokenize_corpus_dict_tweet(df, all_emotions,
                    stop_list=custom_stopwords + punc_custom,
                                        verbose=False)
all_tokens = []
[all_tokens.extend(tokens) for tokens in token_dict.values()]

# get frequency distribution for entire corpus
all_corpus = FreqDist(all_tokens)

corpus_freq_df = pd.DataFrame(all_corpus.most_common(100),
                              columns=['Word','Count'])
corpus_freq_df[:5]
```

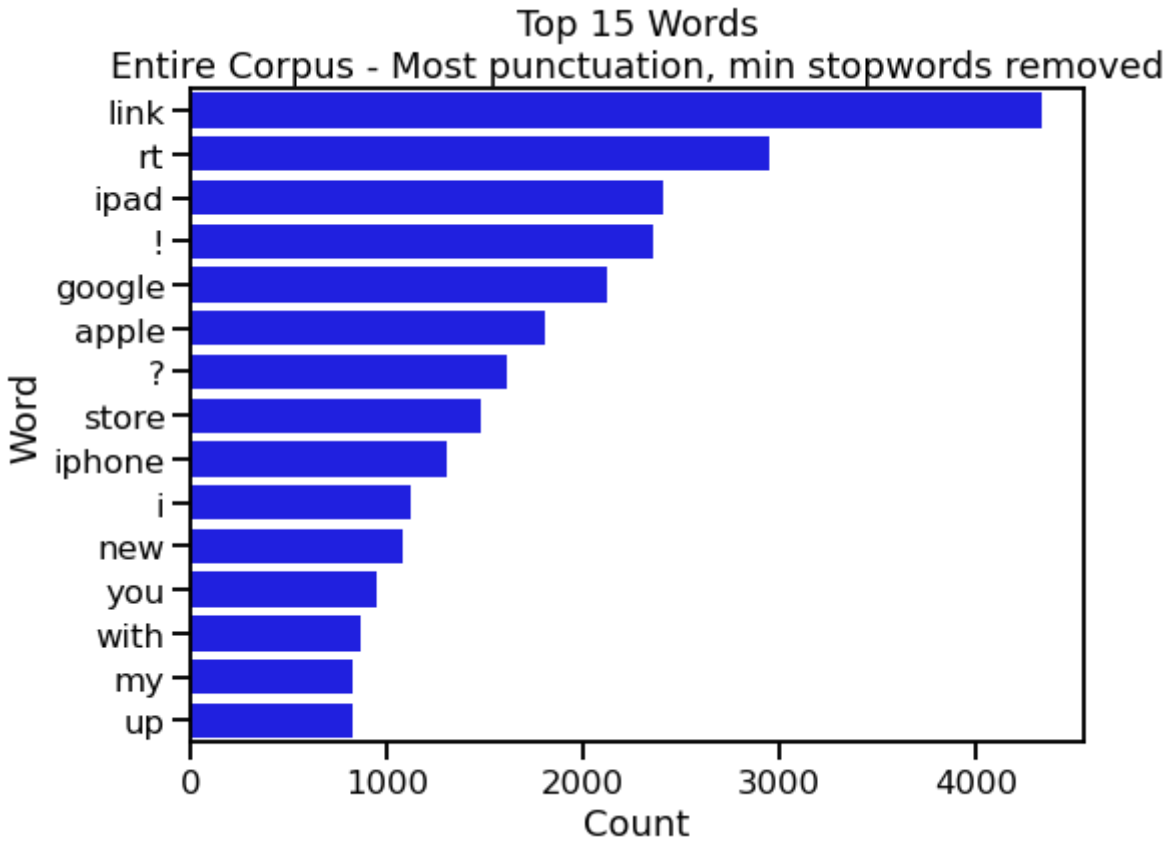
Out[329…

	Word	Count
0	link	4336
1	rt	2948
2	ipad	2407
3	!	2357
4	google	2125

In [330…

```
# Wordcloud for entire corpus, only custom stopwords and punct removed
generate_wordcloud(docs=all_tokens, cmap="Set1",
                   stopwords=None, min_font_size=18)

plot_wordfreqs(corpus_freq_df, 'Word', 'Count', 15,
               "Entire Corpus - Most punctuation, min stopwords removed")
```





NLTK stopwords and punctuation

In [331…

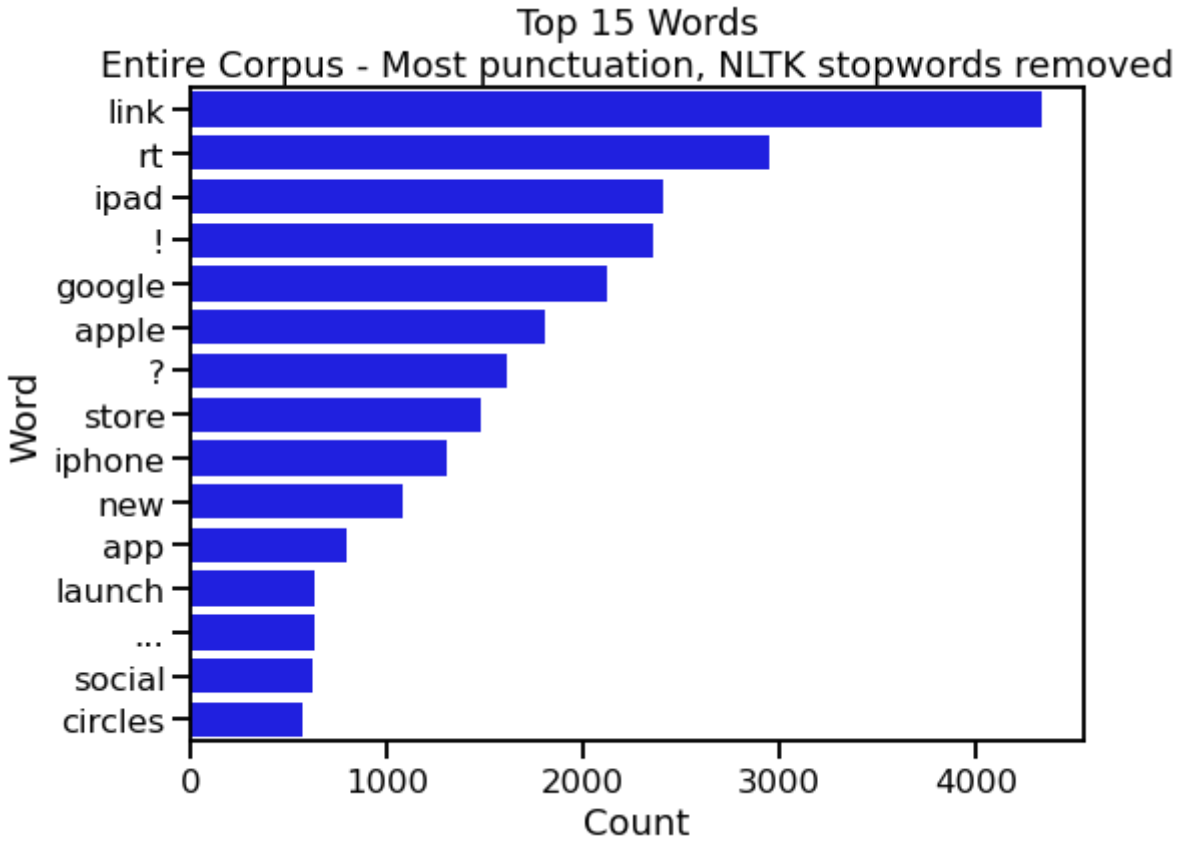
```
# make a new dict with stopwords and punctuation removed
token_dict = tokenize_corpus_dict_tweet(df, all_emotions,
                    stop_list=nltk_stopwords + punc_custom,
                                          verbose=False)
all_tokens = []
[all_tokens.extend(tokens) for tokens in token_dict.values()]

# get frequency distribution for entire corpus
all_corpus = FreqDist(all_tokens)

corpus_freq_df = pd.DataFrame(all_corpus.most_common(100),
                              columns=['Word','Count'])
corpus_freq_df[:5]
```

Out[331…

	Word	Count
0	link	4336
1	rt	2948
2	ipad	2407
3	!	2357
4	google	2125

In [332…

```
generate_wordcloud(docs=all_tokens, cmap="Set1",
                   stopwords=None, min_font_size=18)

plot_wordfreqs(corpus_freq_df, 'Word', 'Count', 15,
               "Entire Corpus - Most punctuation, NLTK stopwords removed")
```



Hmm, so I still see `'link` and `rt` very high on the list, and know that link represents the links I replaced, and rt means a re-tweet.

I'm going to add both of those to my main stopwords lists.

I also see there are a bunch of Google and Apple products as well as brand names. I'm going to create separate lists of the product stopwords, so I can test removing them during modeling or not.

In [333…

```
# create new stopwords list for product names and hashtags
product_stopwords = ["google", "apple", "ipad", "iphone", "android", "ipad2"]
hash_stop = ["#" + word for word in product_stopwords]
product_stopwords.extend(hash_stop)

# extend both existing stopwords lists to include placeholder text link and rt
nltk_stopwords.extend(['link', 'rt'])
custom_stopwords.extend(['link', 'rt'])
```

In [334…

```
# Do all of the tweets contain one of the product stopwords?
# let's look at those that don't contain any of these words (will search substri
df.loc[df['cleaned'].str.contains('|'.join(product_stopwords), case=False)==Fals
```

Out[334…

	tweet_text	product	emotion	cleaned	hashtags	raw_token_count
50	‰ÛÏ@mention {link} <-- HELP ME FORWARD THIS DOC to all Anonymous accounts, techies,& ppl who can help us JAM #libya #SXSW	NaN	No emotion toward brand or product	@mention {link} <-- HELP ME FORWARD THIS DOC to all Anonymous accounts, techies,& ppl who can help us JAM #libya #SXSW	#libya #SXSW	26
51	‰÷¼ WHAT? ‰÷_ {link} ‰ã_ #edchat #musedchat #sxsw #sxswi #classical #newTwitter	NaN	No emotion toward brand or product	WHAT? _ {link} _ #edchat #musedchat #sxsw #sxswi #classical #newTwitter	#edchat #musedchat #sxsw #sxswi #classical #newTwitter	13
52	.@mention @mention on the location-based 'fast, fun and future' - {link} (via @mention #sxsw	NaN	No emotion toward brand or product	.@mention @mention on the location-based 'fast, fun and future' - {link} (via @mention #sxsw	#sxsw	21
65	At #sxsw? @mention / @mention wanna buy you a drink. 7pm at Fado on 4th. {link} Join us!	NaN	No emotion toward brand or product	At #sxsw? @mention / @mention wanna buy you a drink. 7pm at Fado on 4th. {link} Join us!	#sxsw	24
70	Chilcott: @mention #SXSW stand talking with Blogger staff. Too late to win competition for best tweet mentioning @mention So no t-shirt.	NaN	No emotion toward brand or product	Chilcott: @mention #SXSW stand talking with Blogger staff. Too late to win competition for best tweet mentioning @mention So no t-shirt.	#SXSW	24
…	…	…	…	…	…	…

	tweet_text	product	emotion	cleaned	hashtags	raw_token_count
8904	Z6: No News is Good News {link} [codes valid: 4:00-7:59:59p 03/11/11] #infektd #sxsw #zlf	NaN	No emotion toward brand or product	Z6: No News is Good News {link} [codes valid: -7:59:59p /11/11] #infektd #sxsw #zlf	#infektd #sxsw #zlf	23
8908	CLIENT NEWS! @mention Releases "Dope Melodies & Heavy Bass" & Invades #SXSW -> {link}	NaN	No emotion toward brand or product	CLIENT NEWS! @mention Releases "Dope Melodies & Heavy Bass" & Invades #SXSW -> {link}	#SXSW	19
8942	This is my 5th year downloading the #sxsw Music Torrent {link} ALL FREE and LEGAL! Great Music.	NaN	No emotion toward brand or product	This is my 5th year downloading the #sxsw Music Torrent {link} ALL FREE and LEGAL! Great Music.	#sxsw	21
8996	by the way, we're looking for a spanish-speaking trend scout based in Austin -> {link} #sxsw	NaN	No emotion toward brand or product	by the way, we're looking for a spanish-speaking trend scout based in Austin -> {link} #sxsw	#sxsw	19
8998	True story! RT @mention I just rated Amy's Ice Cream 5 stars. @mention "Best ice cream in town!!" {link} #sxsw	NaN	No emotion toward brand or product	True story! RT @mention I just rated Amy's Ice Cream stars. @mention "Best ice cream in town!!" {link} #sxsw	#sxsw	27

787 rows × 6 columns

All stopwords and punctuation

In [335…

```
# make a new dict with stopwords and punctuation removed
token_dict = tokenize_corpus_dict_tweet(df, all_emotions,
        stop_list=nltk_stopwords + punc_custom + product_stopwords,
        verbose=False)

all_tokens = []
[all_tokens.extend(tokens) for tokens in token_dict.values()]

# get frequency distribution for entire corpus
all_corpus = FreqDist(all_tokens)

corpus_freq_df = pd.DataFrame(all_corpus.most_common(100),
                              columns=['Word','Count'])
corpus_freq_df[:5]
```

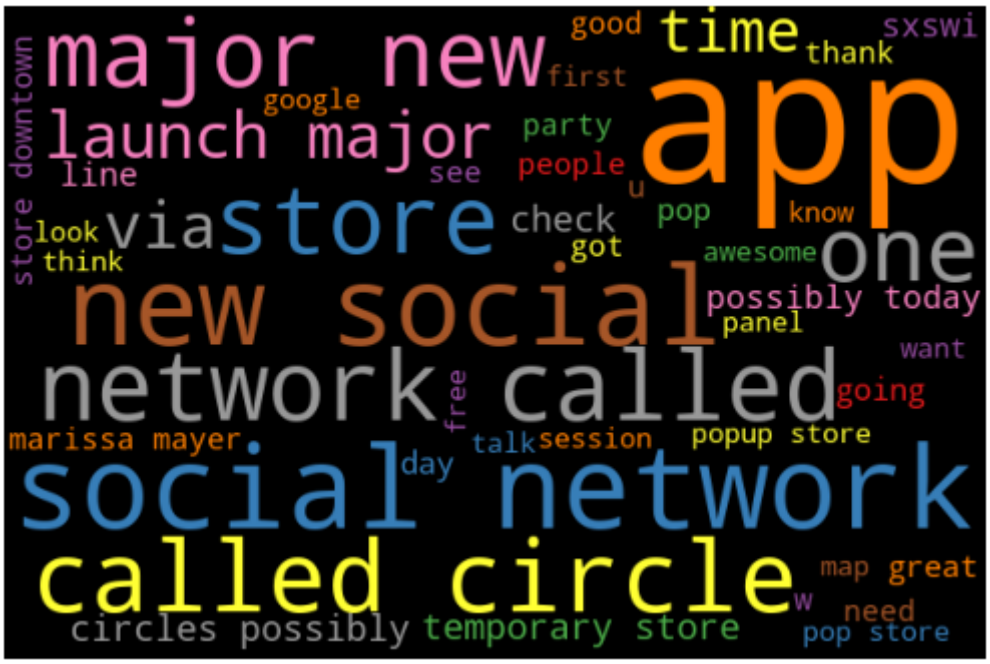
Out[335…

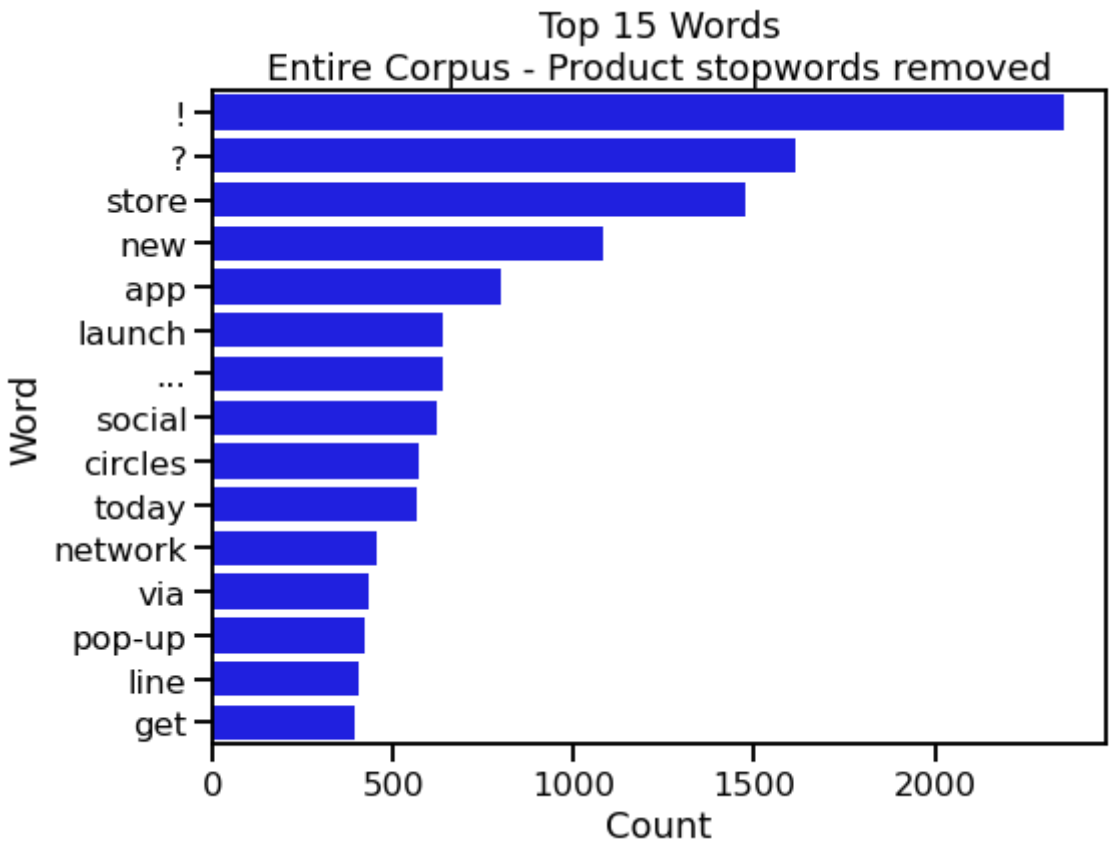
	Word	Count
0	!	2357

	Word	Count
1	?	1613
2	store	1477
3	new	1082
4	app	799

In [336…

```
generate_wordcloud(docs=all_tokens, cmap="Set1",
                   stopwords=None, min_font_size=16)

plot_wordfreqs(corpus_freq_df, 'Word', 'Count', 15,
               "Entire Corpus - Product stopwords removed")
```



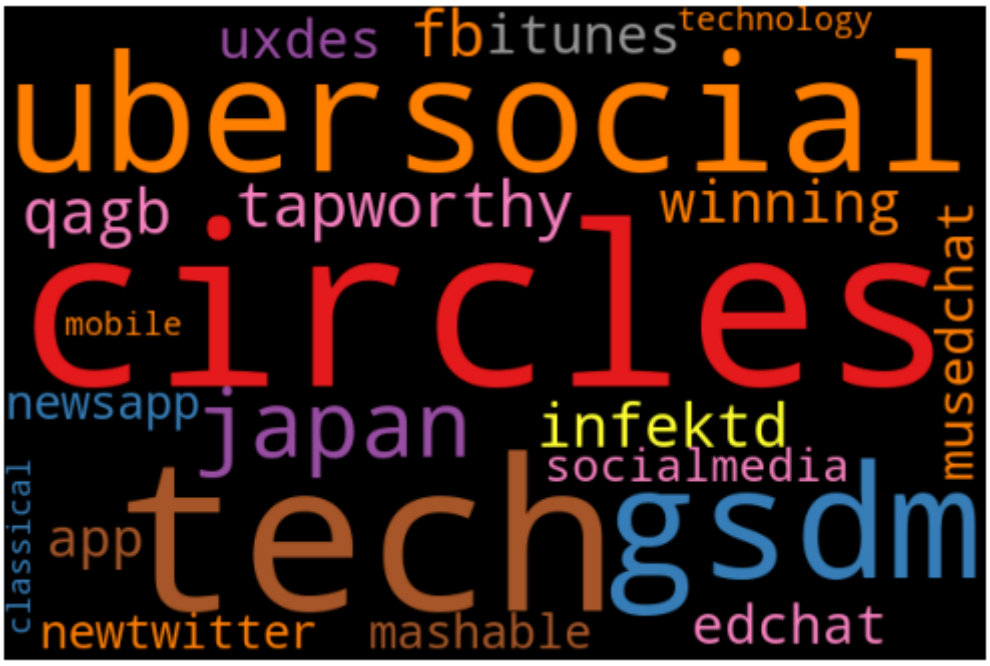


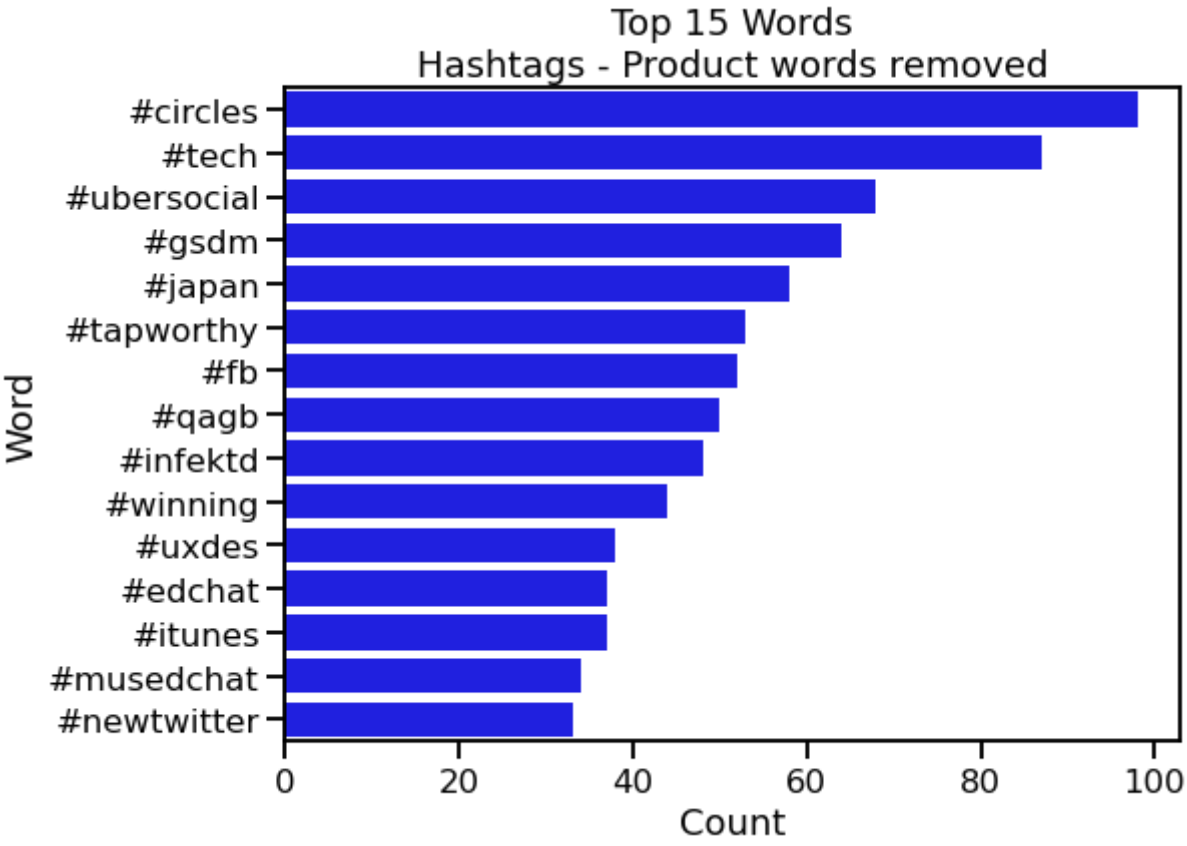
Most common hashtags

In [337…
```
# make a new dict with stopwords and punctuation removed
token_hash_dict = tokenize_corpus_dict_tweet(df, all_emotions,
        stop_list=nltk_stopwords + punc_custom +
            product_stopwords + ['#sxswi', '#austin'],
        verbose=False, doc_col='hashtags')

hash_tokens = []
[hash_tokens.extend(tokens) for tokens in token_hash_dict.values()]

hash_tokens[:5]
```

Out[337… `['#speechtherapy', '#iear', '#edchat', '#asd', '#gdgtlive']`

In [338…
```
# get frequency distribution for entire corpus
hash_corpus = FreqDist(hash_tokens)

hash_freq_df = pd.DataFrame(hash_corpus.most_common(100),
                                columns=['Word','Count'])

generate_wordcloud(docs=hash_tokens, cmap="Set1",
                   stopwords=None, min_font_size=18, n_grams=False)

plot_wordfreqs(hash_freq_df, 'Word', 'Count', 15,
               "Hashtags - Product words removed")
```





Explore Per Class Label

In [339…

```
# Go back to only removing custom punctuation and stopwords
token_dict = tokenize_corpus_dict_tweet(df, all_emotions,
        stop_list=custom_stopwords + punc_custom, verbose=False)

all_tokens = []
[all_tokens.extend(tokens) for tokens in token_dict.values()]

# get frequency distribution for entire corpus
```

```
all_corpus = FreqDist(all_tokens)

corpus_freq_df = pd.DataFrame(all_corpus.most_common(100),
                              columns=['Word','Count'])
corpus_freq_df[:5]
```

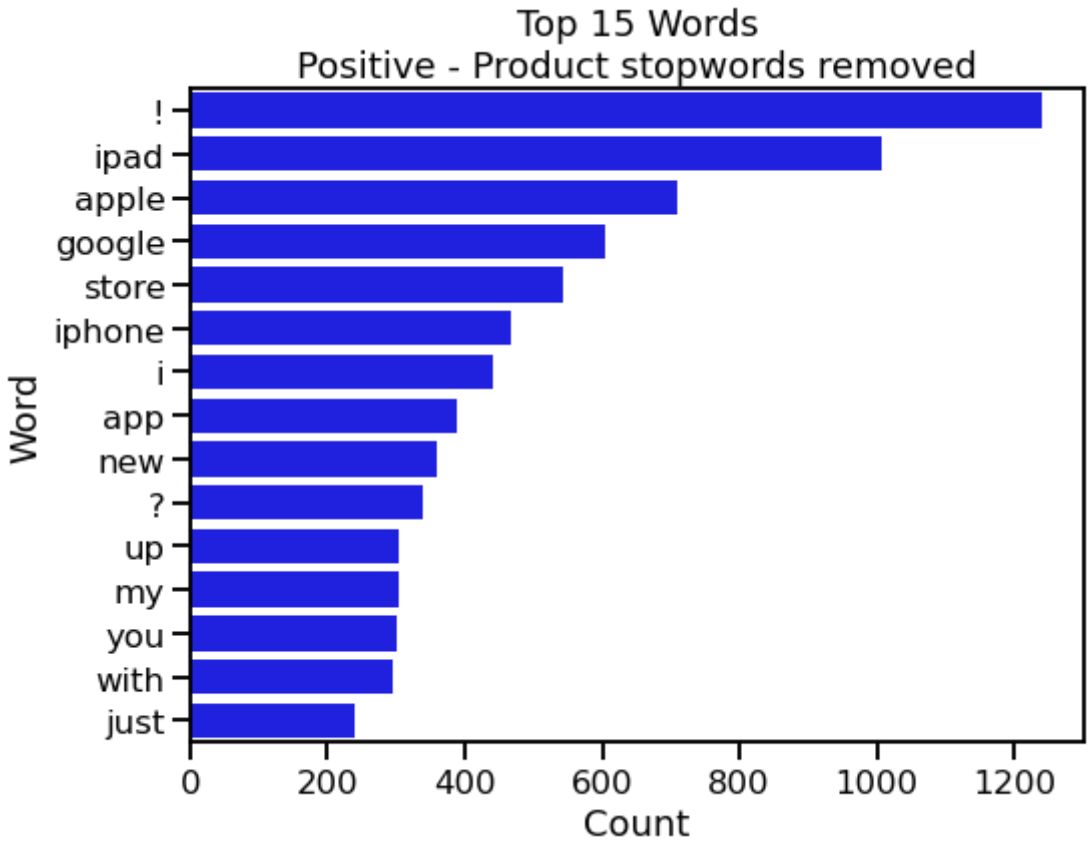
Out[339…

	Word	Count
0	ipad	2407
1	!	2357
2	google	2125
3	apple	1811
4	?	1613

Positive

In [340…

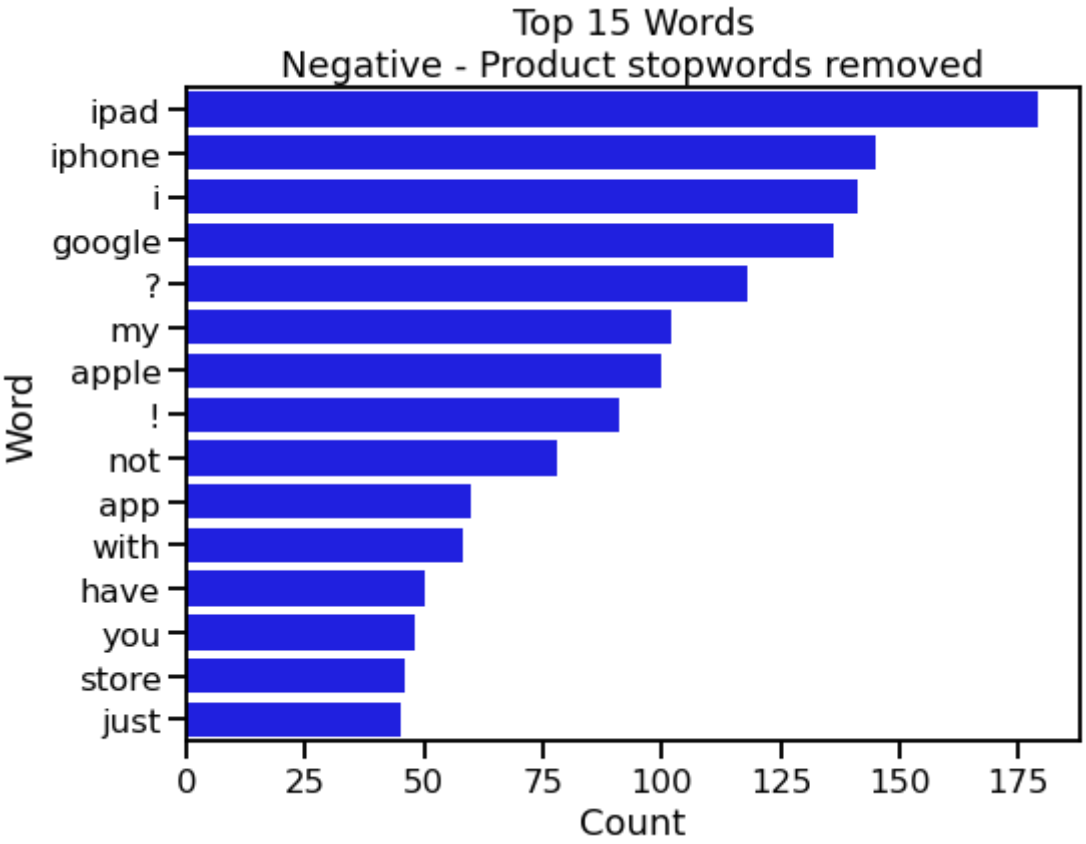
```
# get frequency distribution for positive emotions
pos = FreqDist(token_dict['Positive emotion'])

pos_freq_df = pd.DataFrame(pos.most_common(100),
                              columns=['Word','Count'])

generate_wordcloud(docs=token_dict['Positive emotion'], cmap="Greens",
                   stopwords=None, min_font_size=14)

plot_wordfreqs(pos_freq_df, 'Word', 'Count', 15,
               "Positive - Product stopwords removed")
```



Negative

In [341…

```
# get frequency distribution for negative emotions
neg = FreqDist(token_dict['Negative emotion'])

neg_freq_df = pd.DataFrame(neg.most_common(100),
                                columns=['Word','Count'])

generate_wordcloud(docs=token_dict['Negative emotion'], cmap="OrRd",
                   stopwords=None, min_font_size=16)

plot_wordfreqs(neg_freq_df, 'Word', 'Count', 15,
               "Negative - Product stopwords removed")
```



Comparing the positive and negative classes, `app`, `store`, `new`, and the punctuation `?`, `!`, and `...` are in the top of both.

The positive wordcloud that shows bigrams has `popup store` and `temporary store`, so people seem to be saying good things about that. `Map` and `free` also make an appearance in the positive word cloud.

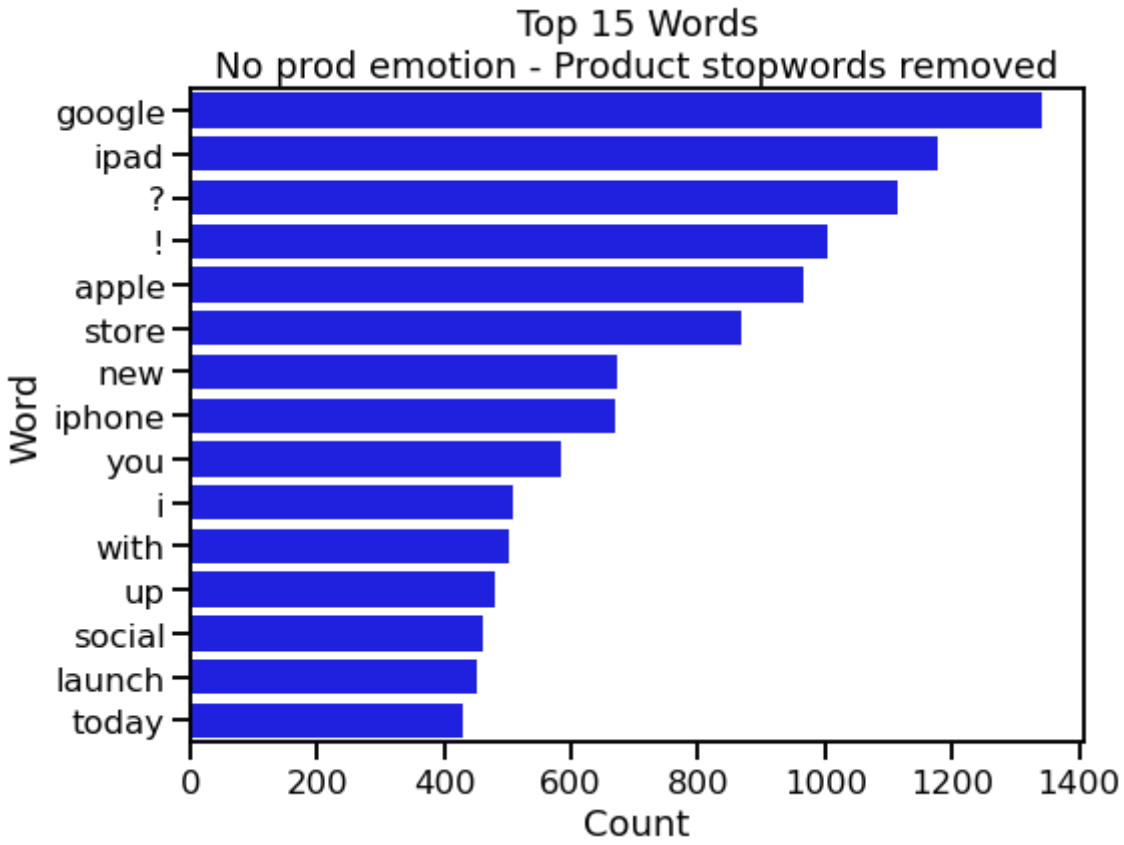
In the negative wordcloud, I see `long`, `line`, `time`, `people`, and `wait`, so I wonder if some of the negative emotions are regarding lines to get into stores or events for the products. I

also see `battery` , `product` , and `design headaches` in the negative wordcloud.

No emotion towards a brand or product

In [342…

```
# get frequency distribution for negative emotions
non = FreqDist(token_dict['No emotion toward brand or product'])

non_freq_df = pd.DataFrame(non.most_common(100),
                           columns=['Word','Count'])

generate_wordcloud(docs=token_dict['No emotion toward brand or product'],
                   cmap="PuBu",
                   stopwords=None, min_font_size=16)

plot_wordfreqs(non_freq_df, 'Word', 'Count', 15,
               "No prod emotion - Product stopwords removed")
```



In the 'No emotion towards a brand or product' class, `major new`, `social network` and `called circle` are top bigrams. A little research turned up that Google+ had a feature called Circles that let you determine who you shared which parts of your feed with, so I suspect at least some of these may be about Google+.

The word `app` is still in the top 15, but in a lower position in the 'No emotion' group than it was in positive and negative. It seems the positive/negative emotions towards brands or products commonly refer to apps.

In [343…

```
df.loc[(df['tweet_text'].str.contains("circle")) &
       (df['emotion']=='No emotion toward brand or product'),
        ['tweet_text']]
```

Out[343…

	tweet_text
158	Anyone at #sxsw or heading to aclu event seen owt to do with google circles then?
311	Those who are tweeting that Google is coming out with Circles platform today are outside of the circle, keep up #itsnot #sxsw
421	Google to debut new selective social network today at #sxsw ? @mention {link} #google #circles
709	.@mention is competing with @mention circles on the SocialFlow board for most resonant topic here. #sxsw {link}
770	Google to Launch Major New Social Network Called Circles, Possibly Today {link} #circles #sxsw
771	Google to Launch Major New Social Network Called Circles, Possibly Today {link} #google #circles #sxsw
819	Google to launch social network "Circles" - denies launch at #sxsw though. {link} #google #circles

	tweet_text
820	Google to launch social network called circles #sxsw RT
825	Is Google launching its own social network? Definitely not at ##sxsw. But rumor still has it...{link} #googlecircles
1043	‰ÛÏ@mention RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles‰Û
1146	Following the buzz? (Hee hee) - Google drip feeding information about Facebook rival: #circles #sxsw
1329	what's up with Google circles? #sxsw
1433	Anonymity: Zuckerberg "wrong" says 4chan's Moot #SXSW {link} VIA @mention Google circles opportunity with profiles?
1642	Cool! RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
1913	I can't wait to see what @mention 's been up to! @mention big news {link} #google #circles #sxsw
2366	#google #circles #sxsw #shice /via @mention {link}
2650	looks like google is publicly denying that "circles" will be presented at #sxsw today.
2673	So #google is going to try #socialnetworking again with #googlecircles to be announced today at #sxsw We shall see...
3835	#Google Circles & crop circles: the anatomy of a #SXSW rumor -{link} by @mention
3868	@mention is that #google circles rumour ??? @mention says We're not launching any products at #SXSW
4075	Will Google Circles take on Facebook? {link} #circles #sxsw
4076	Will Google Circles take on Facebook. {link} #circles #sxsw
4331	Hmm. RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
4454	| @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles | via @mention
4535	does anyone know if google did talk about #circles at #SXSW ?
4603	Interesting news ‰ÛÏ@mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles‰Û
4627	Interesting. Google will launch circles later today where context is added to search #sxsw {link}
4821	#Google to Announce Google Circles Social Network at #SXSW {link} #seo #sem #googlecircles #topnews
5040	RT @mention .@mention is competing with @mention circles on the SocialFlow board for most resonant topic here. #sxsw {link}
5062	RT @mention ‰ÛÏ@mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles‰Û sounds interesting
5217	RT @mention #Google Circles & crop circles: the anatomy of a #SXSW rumor -{link} by @mention
5541	RT @mention Ben benieuwd! RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
5894	RT @mention Google to Launch Major New Social Network Called Circles!!! #google #circles #sxsw {link}

	tweet_text
6121	RT @mention Interesting. Google will launch circles later today where context is added to search #sxsw {link}
6256	RT @mention looks like google is publicly denying that "circles" will be presented at #sxsw today.
6568	RT @mention RT @mention news Google may launch "social circles" social network today {link} #soccomp #sxsw
6579	RT @mention RT @mention RT @mention news Google may launch "social circles" social network today {link} #soccomp #sxsw
6593	RT @mention RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
6594	RT @mention RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
6855	RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
6856	RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles // this is HUGE!
6857	RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles #socbiz
6858	RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
6875	RT @mention We're not launching any products at #SXSW but we're doing plenty else. {link} << no circles or new apps ftrs?
6876	RT @mention We're not launching any products at #SXSW but we're doing plenty else. {link} #googlecircle
6910	RT @mention Will Google Circles take on Facebook? {link} #circles #sxsw
6911	RT @mention Will Google Circles take on Facebook. {link} #circles #sxsw
7012	What is google circles?? ‰ÛÏ@mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw‰Û
7432	Google supposedly announcing new Social Networking service called "circles" today at #sxsw {link}
7564	Google Circles, what is it and will it ever arrive? {link} #google #circles #social #search #sxsw
8453	Google maths of the day: wave + buzz = circles! #google #circles #fail #sxsw
8568	don't distrub my circles - RT @mention We're not launching any products at #SXSW - dancing today {link}
8643	Google launching social network "google circles" @mention #sxsw ?????
8663	Apple popup store line still circles the block. #sxsw
8672	We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
8673	We interrupt your regularly-scheduled #sxsw geek programming with Big news! {link} by @mention #google #circles
8757	#google (via @mention denies it will be launching #circles at #SXSW -hope it fares better than #wave and #buzz when it comes
9003	... or maybe not: {link} #google #circles #sxsw

	tweet_text
9050	Cue the hype RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles
9061	Wave, buzz... RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles

I see that most of the tweets mentioning the word `circle` are talking about a rumored launch of Google Circles. Some I would agree are neutral in sentiment, but many seem to be excited about Circles, so I'm wondering why they were not labeled as Positive? An example is row 1644.

Summary of Exploration

I feel confident about the stopwords and punctuation removal at this point, having a few different versions of lists I can try out to see what works best in modeling.

I see some differences in the most common words between classes, but not a ton. I'm going to have to let the models tell me what else they can glean!

Export data for gridsearching in Colab

To efficiently gridsearch different parameters for different types of models (including preprocessing steps such as stemming and lemmatization) I exported the cleaned dataframe and stopwords lists I had compiled. This allowed me to run gridsearches in Google Colab in parallel with gridsearching other models on my local machine.

In [344…
```
# dump preprocessed df to file, so I can load it up in google colab for
# modeling and gridsearching
joblib.dump(df, f"data/scrubbed_df.joblib.gz")
```

Out[344… `['data/scrubbed_df.joblib.gz']`

In [345…
```
# dump current stopwords lists, which will be used in modeling, out to file
stop_lists = {"custom_stopwords": custom_stopwords,
 "nltk_stopwords": nltk_stopwords,
 "punc_custom": punc_custom,
 "product_stopwords": product_stopwords}

for stop in stop_lists.keys():
    joblib.dump(stop_lists[stop], f"data/{stop}.joblib.gz")
```

Binary Models for Emotion Only

First, I explored models focused only on the tweets that were labeled as having negative or positive emotions about a product or brand, leaving out the ones labeled as having no emotion towards a product or brand.

I wanted to explore the different options for vectorizing and preprocessing the text on a smaller dataset and with a simpler problem before expanding to a multi-class problem with more data, which will take longer to gridsearch and tune.

This binary classifier will also serve as a baseline for the multi-class problem where I will try to predict Positive, Negative, and No sentiment towards a product or brand. I do want to be able to weed out the tweets which are positive and negative towards a brand from those with no emotion towards a brand, but I'm not sure how well a multi-class model will be able to distinguish between all three. Having looked at the tweets themselves, the distinctions are pretty fine grains, and even I disagree with some of the labels which have been assigned. It might be a better approach to use one model to weed out the tweets which don't have a positive or negative emotion towards a brand, then run the remaining tweets through a second model to separate positive from negative.

A few of the options I'd like to try:

- Different versions of stopwords and punctuation removal. All stopwords from the default NLTK list, which I think may contain some words that will be useful, and then also my customized, pared down stopwords list. I also have the separate product stop words list which contains the terms related to the brands and products themselves, which I want to experiment with removing.
- A few different ways to vectorize and generate the frequencies in my Document Term Matrix. Regular `CountVectorizer` with the actual counts, TFIdF normalized frequencies, and also a binary frequency. The sklearn documentation mentions:

  > "...very short texts are likely to have noisy tf–idf values while the binary occurrence info is more stable."

- Unigrams, uni- and bigrams, and just bigrams.

Some initial constants:

- I've decided to use just one set of punctuation to be removed, where I will remove everything except `?`, `!`, and `...`.
- I will experiment stemming and lemmatization, but will start without them first.

I'll use the `TweetTokenizer` from NLTK to perform the tokenization, since it has the option to remove handled, which I would like to do. To enable experimenting with stemming and lemmatization in a gridsearch, I'll create a custom function which uses the TweetTokenizer, and pass to the `tokenizer` step of the sklearn `CountVectorizer` so I can change it up.

Preprocessing for binary modeling

In [346…

```
# check to make sure the text is still aligned in the columns properly
df.loc[df['emotion'].isin(['Positive emotion', 'Negative emotion']),
           ['tweet_text', 'cleaned', 'emotion']]
```

Out[346…

	tweet_text	cleaned	emotion
0	.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.	.@wesley83 I have a 3G iPhone. After hrs tweeting at #RISE_Austin, it was dead! I need to upgrade. Plugin stations at #SXSW.	Negative emotion
1	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW	@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design. Also, they're giving free Ts at #SXSW	Positive emotion
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	@swonderlin Can not wait for #iPad also. They should sale them down at #SXSW.	Positive emotion
3	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	Negative emotion
4	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	Positive emotion
…	…	…	…
9049	@mention your PR guy just convinced me to switch back to iPhone. Great #sxsw coverage. #princess	@mention your PR guy just convinced me to switch back to iPhone. Great #sxsw coverage. #princess	Positive emotion
9051	"papyrus...sort of like the ipad" - nice! Lol! #SXSW Lavelle	"papyrus...sort of like the ipad" - nice! Lol! #SXSW Lavelle	Positive emotion
9052	Diller says Google TV "might be run over by the PlayStation and the Xbox, which are essentially ready today." #sxsw #diller	Diller says Google TV "might be run over by the PlayStation and the Xbox, which are essentially ready today." #sxsw #diller	Negative emotion
9057	I've always used Camera+ for my iPhone b/c it has an image stabilizer mode. Suggestions for an iPad cam app w/ same feature? #SXSW #SXSWi	I've always used Camera+ for my iPhone b/c it has an image stabilizer mode. Suggestions for an iPad cam app w/ same feature? #SXSW #SXSWi	Positive emotion
9060	Ipad everywhere. #SXSW {link}	Ipad everywhere. #SXSW {link}	Positive emotion

3537 rows × 3 columns

In [347…
```
df['emotion'].value_counts()
```

Out[347…
```
No emotion toward brand or product    5372
Positive emotion                      2968
Negative emotion                       569
I can't tell                           156
Name: emotion, dtype: int64
```

In [348…
```
# create X and y based on just the positive and negative emotions
X = df.loc[df['emotion'].isin(['Positive emotion', 'Negative emotion']),
           'cleaned']
y = df.loc[df['emotion'].isin(['Positive emotion', 'Negative emotion']),
           'emotion']
```

In [349…
```
X.head()
```

```
0                  .@wesley83 I have a 3G iPhone. After hrs tweeting at #RISE_A
```

Out[349…
```
ustin, it was dead! I need to upgrade. Plugin stations at #SXSW.
1    @jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely
appreciate for its design. Also, they're giving free Ts at #SXSW
2                                                                  @swonderlin C
an not wait for #iPad also. They should sale them down at #SXSW.
3                                                                @sxsw I hope this
year's festival isn't as crashy as this year's iPhone app. #sxsw
4                @sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Ti
m O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)
Name: cleaned, dtype: object
```

In [350…
```
# convert class labels to binary
y = y.map(lambda x: 1 if x=="Negative emotion" else 0)
y.value_counts()
```

Out[350…
```
0    2968
1     569
Name: emotion, dtype: int64
```

In [351…
```
# Save label translation for later: 1 is negative, 0 is positive
class_labels = ['Positive', 'Negative']
```

In [352…
```
# train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    stratify=y)
print(len(X_train))
print(len(y_train))
print(len(X_test))
print(len(y_test))
```

```
2829
2829
708
708
```

In [353…
```
print(X_train.shape)
print(y_train.shape)
```

```
(2829,)
(2829,)
```

In [354…
```
y_train.value_counts(normalize=True)
```

Out[354…
```
0    0.839166
1    0.160834
Name: emotion, dtype: float64
```

In [355…
```
y_test.value_counts(normalize=True)
```

Out[355…
```
0    0.838983
1    0.161017
Name: emotion, dtype: float64
```

In [356…
```
def eval_clf_model(clf, X_test, y_test, X_train, y_train, score='macro',
                   reports=True, labels=['Class 0', 'Class 1'],
                   normalize_cm='true'):
    """Shows metrics and plots visualizations to interpret classifier model
    performance.

    ***
    Args
```

```
    clf: classifier model to evaluate (or fit pipeline with a clf model as
    the last step)

    X_test: dataframe of test predictors

    y_test: dataframe of true target values

    X_train: dataframe (optional). Default is None. Provide training data if
    you want to evaluate performance on train versus test; otherwise only
    test performance is evaluated.

    y_train: dataframe (optional). Default is None. Provide training data if
    you want to evaluate performance on train versus test; otherwise only
    test performance is evaluated.

    score: string (optional). Default is `std` to return standard F1, accuracy,
    and recall scores. Use `macro` to return macro F1 and recall, and balanced
    accuracy. Scores are always returned, regardles of `reports` param.

    reports: boolean (optional). Default is True. Set to False to return only
    scores, not actual classification reports.

    labels: list (optional). Provide a list of labels for the target class.

    normalize_cm: string, default `true`. Setting for whetherand how to
    normalize the confusion matrix. See sklearn documentation for options.
    """
    multi = True if len(labels) > 2 else False

    spacer = '*'*30

    # Get predictions 1 time only, since they will be used in a few spots
    test_preds = clf.predict(X_test)
    train_preds = clf.predict(X_train)

    #if multi:
    #    test_predict_proba = clf.predict_proba(X_test)

    # print classification reports
    if reports:
        print(spacer + ' Training Data ' + spacer)
        print(metrics.classification_report(y_train, train_preds))
        print()
        print(spacer + ' Test Data ' + spacer)
        print(metrics.classification_report(y_test, test_preds))
        print()

    # print scores from train and test next to each other for easy comparison
    print(spacer + ' Training Scores ' + spacer)

    # Train
    if score == 'std':
        train_f1 = np.round(metrics.f1_score(y_train, train_preds), 4)
        print(f"                  Training F1 = {train_f1}")
        train_r = np.round(metrics.recall_score(y_train, train_preds), 4)
        print(f"              Training Recall = {train_r}")
        train_acc = np.round(metrics.accuracy_score(y_train, train_preds), 4)
        print(f"            Training Accuracy = {train_acc}")
    elif score == 'macro':
        train_f1m = np.round(metrics.f1_score(y_train, train_preds, average='mac
        print(f"            Training Macro F1 = {train_f1m}")
```

```
    train_rm = np.round(metrics.recall_score(y_train, train_preds, average='
    print(f"        Training Macro Recall = {train_rm}")
    train_accbal = np.round(metrics.balanced_accuracy_score(y_train, train_p
    print(f"   Training Balanced Accuracy = {train_accbal}")
print()
print(spacer + ' Test Scores ' + spacer)

#Test
if score == 'std':
    test_f1 = np.round(metrics.f1_score(y_test, test_preds), 4)
    print(f"                      Test F1 = {test_f1}")
    test_r = np.round(metrics.recall_score(y_test, test_preds), 4)
    print(f"                  Test Recall = {test_r}")
    test_acc = np.round(metrics.accuracy_score(y_test, test_preds), 4)
    print(f"                Test Accuracy = {test_acc}")

elif score == 'macro':
    test_f1m = np.round(metrics.f1_score(y_test, test_preds, average='macro'
    print(f"                Test Macro F1 = {test_f1m}")
    test_rm = np.round(metrics.recall_score(y_test, test_preds, average='mac
    print(f"            Test Macro Recall = {test_rm}")
    test_accbal = np.round(metrics.balanced_accuracy_score(y_test, test_pred
    print(f"       Test Balanced Accuracy = {test_accbal}")
print()
print(spacer + ' Differences ' + spacer)

#Diffs
if score == 'std':
    print(f"               Train-Test F1 Diff = {test_f1 - train_f1}")
    print(f"           Train-Test Recall Diff = {test_r - train_r}")
    print(f"         Train-Test Accuracy Diff = {test_acc - train_acc}")
elif score == 'macro':
    print(f"         Train-Test Macro F1 Diff = {test_f1m - train_f1m}")
    print(f"     Train-Test Macro Recall Diff = {test_rm - train_rm}")
    print(f"Train-Test Balanced Accuracy Diff = {test_accbal - train_accbal}

print()
print(spacer + ' Graphs for Test ' + spacer)

# plot graphs

if not multi:
    auc = np.round(metrics.roc_auc_score(y_test, test_preds), 2)

    ap = np.round(metrics.average_precision_score(y_test, test_preds), 2)

    fig, [ax1, ax2, ax3] = plt.subplots(figsize=[10, 3], nrows=1, ncols=3)
    plt.tight_layout(pad=2.5)
    metrics.plot_confusion_matrix(clf, X_test, y_test,
            normalize=normalize_cm, display_labels=labels,
                                  cmap='Reds', ax=ax1)
    metrics.plot_roc_curve(clf, X_test, y_test, ax=ax2)
    ax2.legend(loc='best', fontsize='small', labels=[f'AUC: {auc}'])

    metrics.plot_precision_recall_curve(clf, X_test, y_test, ax=ax3)
    ax3.legend(loc='best', fontsize='small')
    ax3.legend(loc='best', fontsize='small', labels=[f'AP: {ap}'])
    plt.show();

#if multi-class, just plot confusion matrix
else:
```

```
        fig, ax1 = plt.subplots(figsize=[6, 4])
        plt.tight_layout(pad=2.5)
        metrics.plot_confusion_matrix(clf, X_test, y_test,
                normalize=normalize_cm, display_labels=labels, cmap='Reds',
                                      ax=ax1)
        plt.show();


    return None
```

In [357…

```
# Path I'm going to use to save best estimators from gridsearching
save_path = "models/"
```

In [358…

```
def clf_gridsearch_wpipe(clf_pipe, grid_params, X_train, y_train, X_test, y_test
                    class_labels, file_name, save_path,
                    scoring='recall_macro', n_jobs=-1, verbose=True,
                    normalize_cm='true'):
    """
    file_name used in exporting best estimator to file
    """
    gs = GridSearchCV(clf_pipe, grid_params, n_jobs=n_jobs, verbose=verbose,
                        scoring=scoring)

    # run the gridsearch
    gs.fit(X_train, y_train)

    # print best estimator params and score
    print(gs.best_estimator_)
    print(gs.best_score_)

    # dump out best estimator and gs object to gdrive
    joblib.dump(gs.best_estimator_.named_steps['clf'],
                f"{save_path}BestEst_{file_name}.joblib.gz")
    print()
    print(f"Saved best estimator to: {save_path}BestEst_{file_name}.joblib.gz")
    joblib.dump(gs, f"{save_path}GSObject_{file_name}.joblib.gz")
    print()
    print(f"Saved GridSearch object to: {save_path}GSObject_{file_name}.joblib.g

    # print the classifier model report
    eval_clf_model(gs, X_test, y_test, X_train, y_train, labels=class_labels,
                  normalize_cm=normalize_cm)

    return None
```

Build modeling pipeline

In [359…

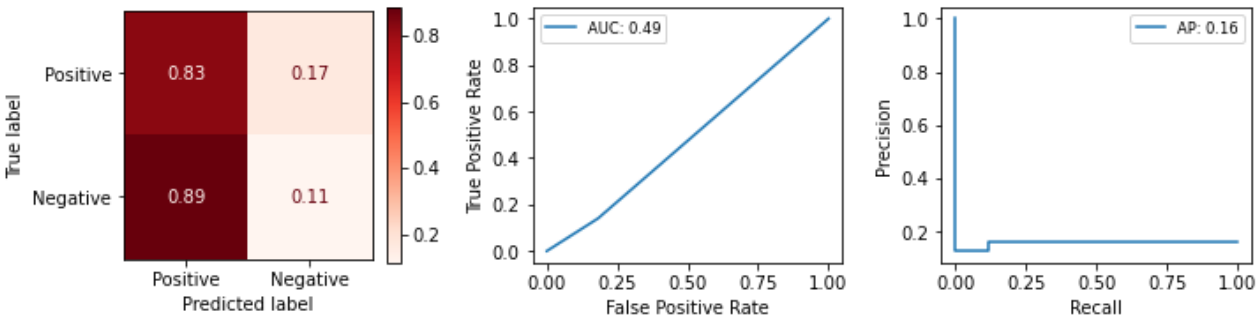
```
# Will use TweetTokenizer with strip_handles=True to tokenize and lowercase
# for now, we're not going to be doing any stemming or lemmatization,
# so I'll just use the class as-is
tweettokenizer = TweetTokenizer(preserve_case=False, strip_handles=True)

# pre-processing pipeline to transform into vectors
prep_pipe = Pipeline([
    ('vect', CountVectorizer(tokenizer=tweettokenizer.tokenize)),
    ('trans', TfidfTransformer())
])
```

Baseline Dummy Classifier

In [360…

```
# Let's make a baseline classifier using the dummy model
clf_pipe = Pipeline([
    ('prep', prep_pipe),
    ('clf', DummyClassifier(strategy='stratified'))
])

clf_pipe.fit(X_train, y_train)

eval_clf_model(clf_pipe, X_test, y_test, X_train, y_train,
               labels=class_labels)
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.84      0.84      0.84      2374
           1       0.18      0.18      0.18       455

    accuracy                           0.73      2829
   macro avg       0.51      0.51      0.51      2829
weighted avg       0.73      0.73      0.73      2829


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.84      0.84      0.84       594
           1       0.15      0.15      0.15       114

    accuracy                           0.73       708
   macro avg       0.49      0.49      0.49       708
weighted avg       0.73      0.73      0.73       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.5085
         Training Macro Recall = 0.5085
    Training Balanced Accuracy = 0.5085

****************************** Test Scores ******************************
                Test Macro F1 = 0.4937
             Test Macro Recall = 0.4938
        Test Balanced Accuracy = 0.4938

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.014799999999999924
       Train-Test Macro Recall Diff = -0.014699999999999935
Train-Test Balanced Accuracy Diff = -0.014699999999999935

****************************** Graphs for Test ******************************
```



Testing Freq Options on Multinomial Bayes Models

I'm going to run a few models using a Multinomial Bayes classifier, with minimal processing.

All of the models below are using the preprocessed documents (non-ASCII characters and numbers removed), with Twitter handles removed and lowercasing performed by the TweetTokenizer.

I'm not removing stopwords or punctuation yet for these baseline models.

I'm also not experimenting with different sizes of n-grams.

The main thing I do want to try to determine is which method of calculating the Document Term Matrix is best for this text. Regular `CountVectorizer` with the raw counts, binary count, or TF-IDF normalized. The sklearn documentation mentions:

> "...very short texts are likely to have noisy tf–idf values while the binary occurrence info is more stable."

Rather than gridsearch through the different types of vectorizers, which will x3 my computational load and time, I'd prefer to settle on one of these vectorizing options up front and use gridsearch for other parameters and processing options such as n-grams and which stop-words to remove.

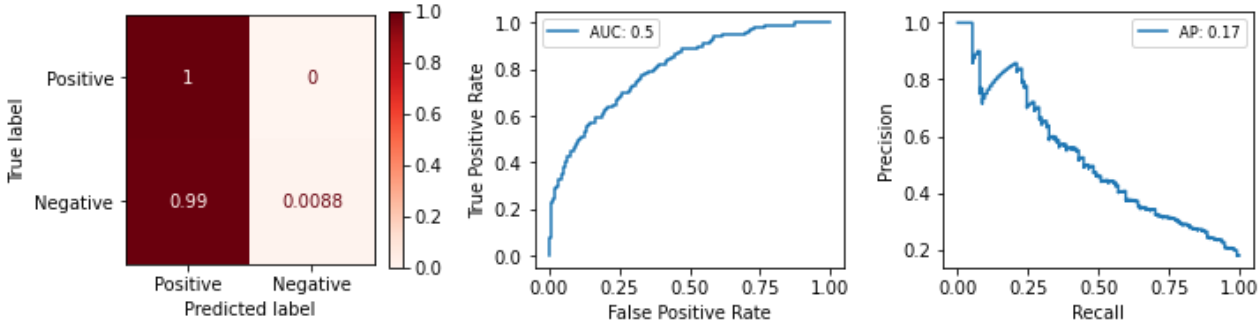
In [361…
```
# modling pipeline with preprocessing and model built in
clf_pipe = Pipeline([
    ('prep', prep_pipe),
    ('clf', MultinomialNB())
])
```

Tfidf Standardized Doc Term Matrix

In [362…
```
clf_pipe.fit(X_train, y_train)

eval_clf_model(clf_pipe, X_test, y_test, X_train, y_train, labels=class_labels)
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.85      1.00      0.92      2374
           1       1.00      0.05      0.10       455

    accuracy                           0.85      2829
   macro avg       0.92      0.53      0.51      2829
weighted avg       0.87      0.85      0.79      2829


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       594
           1       1.00      0.01      0.02       114

    accuracy                           0.84       708
   macro avg       0.92      0.50      0.47       708
weighted avg       0.87      0.84      0.77       708
```

```
****************************** Training Scores ******************************
            Training Macro F1 = 0.5106
         Training Macro Recall = 0.5275
    Training Balanced Accuracy = 0.5275

****************************** Test Scores ******************************
                Test Macro F1 = 0.4653
             Test Macro Recall = 0.5044
        Test Balanced Accuracy = 0.5044

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.04530000000000006
       Train-Test Macro Recall Diff = -0.02310000000000001
Train-Test Balanced Accuracy Diff = -0.02310000000000001

****************************** Graphs for Test ******************************
```



These results are actually worse than the dummy classifier; it's just predicting positive for almost everything.

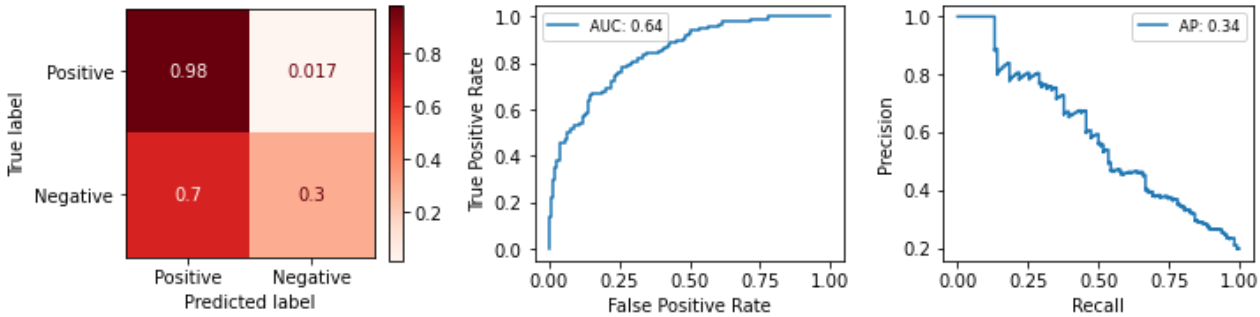
Counted Doc Term Matrix

In [363…

```
# Since my pipeline includes the Tfidf transformer, I'm explicitly setting
# that pipeline step to be passthrough here to remove it
clf_pipe.set_params(prep__vect__binary=False)
clf_pipe.set_params(prep__trans='passthrough')

clf_pipe.fit(X_train, y_train)

eval_clf_model(clf_pipe, X_test, y_test, X_train, y_train, labels=class_labels)
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      2374
           1       0.92      0.72      0.81       455

    accuracy                           0.95      2829
   macro avg       0.94      0.86      0.89      2829
weighted avg       0.95      0.95      0.94      2829


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.88      0.98      0.93       594
           1       0.77      0.30      0.43       114

    accuracy                           0.87       708
   macro avg       0.83      0.64      0.68       708
```

```
weighted avg       0.86      0.87      0.85       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.8899
         Training Macro Recall = 0.8559
    Training Balanced Accuracy = 0.8559

****************************** Test Scores ******************************
                Test Macro F1 = 0.6794
             Test Macro Recall = 0.6407
        Test Balanced Accuracy = 0.6407

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.21050000000000002
       Train-Test Macro Recall Diff = -0.21519999999999995
Train-Test Balanced Accuracy Diff = -0.21519999999999995

****************************** Graphs for Test ******************************
```



Better than Tfidf, but still not great on test, with only 32% recall on my target class of Negative.

It's also looking pretty overfit to the training data.

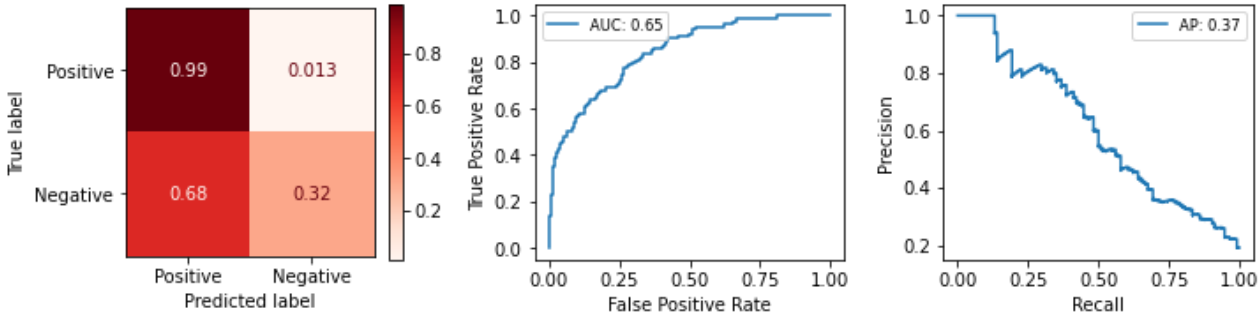
Binary Doc Term Matrix

In [364…

```
# Let's try the binary representation instead of raw counts
clf_pipe.set_params(prep__vect__binary=True)
clf_pipe.set_params(prep__trans='passthrough')

clf_pipe.fit(X_train, y_train)

eval_clf_model(clf_pipe, X_test, y_test, X_train, y_train, labels=class_labels)
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      2374
           1       0.93      0.72      0.81       455

    accuracy                           0.95      2829
   macro avg       0.94      0.85      0.89      2829
weighted avg       0.95      0.95      0.94      2829


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.88      0.99      0.93       594
           1       0.82      0.32      0.46       114

    accuracy                           0.88       708
```

```
   macro avg       0.85      0.65      0.69       708
weighted avg       0.87      0.88      0.86       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.8897
         Training Macro Recall = 0.855
    Training Balanced Accuracy = 0.855

****************************** Test Scores **********************************
                Test Macro F1 = 0.6937
             Test Macro Recall = 0.6512
        Test Balanced Accuracy = 0.6512

****************************** Differences **********************************
          Train-Test Macro F1 Diff = -0.19600000000000006
       Train-Test Macro Recall Diff = -0.20379999999999998
Train-Test Balanced Accuracy Diff = -0.20379999999999998

****************************** Graphs for Test ******************************
```



So testing out the different ways I could calculate frequency in the document term matrix, Tfidf performed the poorest on this test where I've done minimal processing (no stopwords or punctuation removal, for instance).

The performance difference between binary and count isn't much.

Initially, I excluded Tfidf frequency from my modeling based on this test without stopwords and punctuation removed. However, when I tried including it in a model with tuned parameters, including removing stopwords and punctuation, it actually performed better in some cases. I revised my approach, and went back and added Tfidf transformation as a step in gridsearching along with other parameters. Although Tfidf performed the worst on minimally processed documents, it sometimes worked better in conjunction with other preprocessing steps and model parameters.

GridSearchCVs: no stemming or lemmatization

First, I'm going to search through params without applying any stemming or lemmatization, but removing stop words and punctuation.

In [365…

```
# set this variable equal to True to re-run the gridsearches
rerun_grid = False
```

I used this same set of vectorization and preprocessing parameters to gridsearch all of the models, changing only the model-specific params, and updating the tokenizer to include

stemming and lemmatization or not.

I initially tried to let the best preprocessing params from one gridsearch inform what should be included/excluded in other gridsearches that used different classifiers. However, upon further testing, I found this was not a sound approach, and decided to do an exhaustive gridsearch for each classifier and modeling scenario.

In [366…
```
# Params that will apply to the vectorizer regardless of how the
# document term matrix is constructed
common_params = {"vect__stop_words": [custom_stopwords + punc_custom,
            nltk_stopwords + punc_custom,
            product_stopwords + custom_stopwords + punc_custom,
            punc_custom + product_stopwords + nltk_stopwords],
    "vect__ngram_range": [(1,1), (1,2), (2,2)],
    "vect__max_features": [None, 1000]}

# Params for the binary frequency, where we don't want to bother with a Tfidf
binary_params = {
    "vect__binary": [True],
    "trans": ['passthrough']}

# params for non-binary frequency, where we want to test both with and without
# Tfidf
count_params = {
    "vect__binary": [False],
    "trans": [TfidfTransformer(), 'passthrough']}
```

Multinomial Bayes

In [367…
```
if rerun_grid:
    # Add Multinomial Bayes as the classifier model
    clf_pipe = Pipeline([
        ('prep', prep_pipe),
        ('clf', MultinomialNB())
    ])

    # classifier-specific params for MNB
    clf_params = {"clf__fit_prior": [True, False]}

    # create grid params for MNB
    grid_params = [{**common_params, **binary_params, **clf_params},
                   {**common_params, **count_params, **clf_params}]

    model_name = "MNB_binary_nostemlem"

    # gridsearch MNB
    clf_gridsearch_wpipe(clf_pipe, grid_params, X_train, y_train, X_test, y_test
                         class_labels, file_name=model_name, save_path=save_path
                         scoring='recall_macro')
```

The gridsearch above was actually executed in Google Colab, from which I exported the best estimator and GS object to file. I'll load back in here to show the results.

Since colab is running a slightly different version of python, I'll load in the best estimator and then create a new pipeline using the best params. This will also make sure that I'm re-fitting the model to the current train split.

In [368…

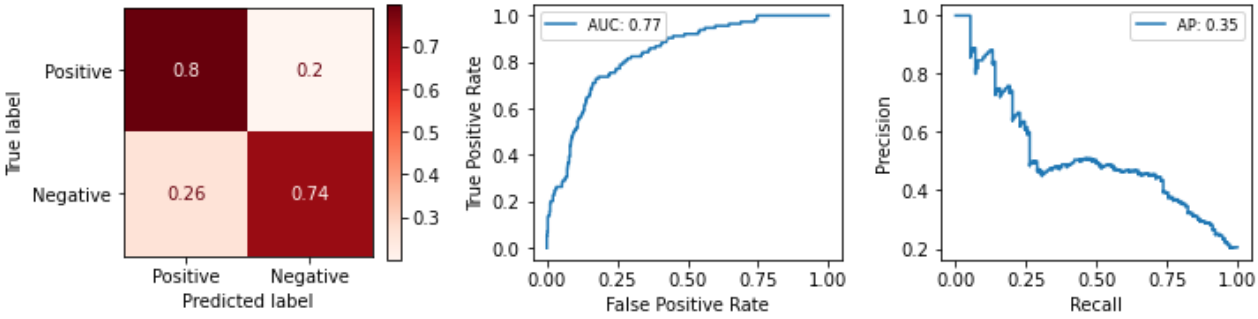
```
def load_rebuild_eval_bestpipe(gsfile_name, X_train, y_train, X_test, y_test,
                               class_labels, load_path=''):
    """
    Loads and rebuilds the best pipeline (including estimator) from a
    GridSearch object that was dumped to file using `joblib`. Fits on
    X_train and y_train, and runs classifier evaluation function to show
    model performance.

    Returns new pipeline object built using the best params from the gridsearch,
    and the gridsearch object itself, so it can be queried to show its best
    score.

    `load_path` should be the path to the gridsearch file to load in, if not
    in current directory.
    """

    gs = joblib.load(load_path + gsfile_name)

    pipe = Pipeline(gs.best_estimator_.get_params()['steps'])

    print(gs.best_estimator_.get_params()['steps'])
    print()
    print(f"Best score from GS-CV: {np.round(gs.best_score_, 3)}")
    print()

    pipe.fit(X_train, y_train)

    eval_clf_model(pipe, X_test, y_test, X_train, y_train,
                   labels=class_labels)

    return pipe, gs
```

In [369…

```
# load and eval pipeline with best params from colab gridsearch
best_MNB_binary_nostemlem_pipe, best_MNB_binary_nostemlem_gs = \
load_rebuild_eval_bestpipe('GSObject_MNB_binary_nostemlem.joblib.gz',
                           X_train, y_train, X_test, y_test,
                           class_labels, load_path=save_path)
```

```
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator CountVectorizer fro
m version 0.22.2.post1 when using version 0.23.2. This might lead to breaking co
de or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator TfidfTransformer fr
om version 0.22.2.post1 when using version 0.23.2. This might lead to breaking c
ode or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator Pipeline from versi
on 0.22.2.post1 when using version 0.23.2. This might lead to breaking code or i
nvalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator MultinomialNB from
version 0.22.2.post1 when using version 0.23.2. This might lead to breaking code
or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator GridSearchCV from v
ersion 0.22.2.post1 when using version 0.23.2. This might lead to breaking code
or invalid results. Use at your own risk.
```

```
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/feature_extraction/text.py:383: UserWarning: Your stop_words may be incons
istent with your preprocessing. Tokenizing the stop words generated tokens ['2']
not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
[('prep', Pipeline(steps=[('vect',
                 CountVectorizer(binary=True, max_features=1000,
                                 stop_words=['google', 'apple', 'ipad',
                                             'iphone', 'android', 'ipad2',
                                             '#google', '#apple', '#ipad',
                                             '#iphone', '#android', '#ipad2',
                                             'a', 'an', 'and', 'am', 'are',
                                             'as', 'at', 'be', 'by', 'for',
                                             'from', 'if', 'in', 'is', 'into',
                                             'it', "it's", 'its', ...],
                                 tokenizer=<bound method TweetTokenizer.tokenize
of <nltk.tokenize.casual.TweetTokenizer object at 0x7fd208ef09d0>>)),
                ('trans', 'passthrough')])), ('clf', MultinomialNB(fit_prior=Fal
se))]

Best score from GS-CV: 0.757

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.97      0.83      0.90      2374
           1       0.50      0.87      0.63       455

    accuracy                           0.84      2829
   macro avg       0.73      0.85      0.76      2829
weighted avg       0.89      0.84      0.85      2829


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.94      0.80      0.86       594
           1       0.41      0.74      0.53       114

    accuracy                           0.79       708
   macro avg       0.68      0.77      0.70       708
weighted avg       0.86      0.79      0.81       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.7643
         Training Macro Recall = 0.8494
   Training Balanced Accuracy = 0.8494

****************************** Test Scores ******************************
                Test Macro F1 = 0.6972
             Test Macro Recall = 0.7683
        Test Balanced Accuracy = 0.7683

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.06709999999999994
       Train-Test Macro Recall Diff = -0.08110000000000006
Train-Test Balanced Accuracy Diff = -0.08110000000000006

****************************** Graphs for Test ******************************
```



Yeah, that looks much more similar to the results in the colab environment. I think I can trust the best params from the colab gridsearch, but need to rebuild and fit the estimator to actually evaluate performance and coefficients.

Random Forest

In [370…

```
if rerun_grid:
    # Subbing Random Forest for the classifier
    clf_pipe = Pipeline([
        ('prep', prep_pipe),
        ('clf', RandomForestClassifier(class_weight='balanced'))
    ])

    # classifier-specific params for RF
    clf_params = {"clf__criterion": ['gini', 'entropy'],
        "clf__max_depth": [5, 10, 50]}

    # create grid params for RF
    grid_params = [{**common_params, **binary_params, **clf_params},
                   {**common_params, **count_params, **clf_params}]

    model_name = "RF_binary_nostemlem"

    # run gridsearch for RF
    clf_gridsearch_wpipe(clf_pipe, grid_params, X_train, y_train, X_test, y_test
                         class_labels, file_name=model_name, save_path=save_path
                         scoring='recall_macro')
```

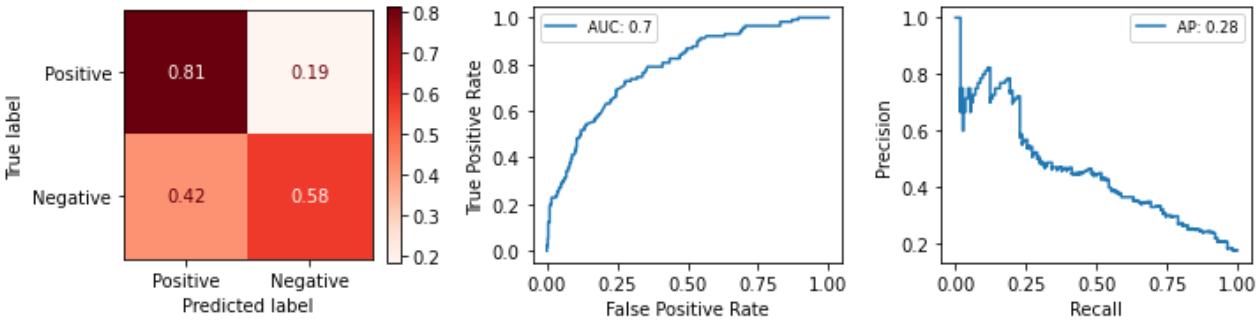
In [371…

```
# load and eval pipeline with best params from colab gridsearch
best_RF_binary_nostemlem_pipe, best_RF_binary_nostemlem_gs = \
load_rebuild_eval_bestpipe('GSObject_RF_binary_nostemlem.joblib.gz',
                           X_train, y_train, X_test, y_test,
                           class_labels, load_path=save_path)
```

```
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator CountVectorizer fro
m version 0.22.2.post1 when using version 0.23.2. This might lead to breaking co
de or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator TfidfTransformer fr
om version 0.22.2.post1 when using version 0.23.2. This might lead to breaking c
ode or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator Pipeline from versi
on 0.22.2.post1 when using version 0.23.2. This might lead to breaking code or i
nvalid results. Use at your own risk.
  warnings.warn(
```

```
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator DecisionTreeClassif
ier from version 0.22.2.post1 when using version 0.23.2. This might lead to brea
king code or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator RandomForestClassif
ier from version 0.22.2.post1 when using version 0.23.2. This might lead to brea
king code or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator GridSearchCV from v
ersion 0.22.2.post1 when using version 0.23.2. This might lead to breaking code
or invalid results. Use at your own risk.
  warnings.warn(
[('prep', Pipeline(steps=[('vect',
                 CountVectorizer(binary=True, max_features=1000,
                                 stop_words=['a', 'an', 'and', 'am', 'are',
                                             'as', 'at', 'be', 'by', 'for',
                                             'from', 'if', 'in', 'is', 'into',
                                             'it', "it's", 'its', 'itself',
                                             'of', 'on', 'or', 'than', 'that',
                                             'the', 'to', 'austin', 'sxsw',
                                             '#sxsw', 'link', ...],
                                 tokenizer=<bound method TweetTokenizer.tokenize
of <nltk.tokenize.casual.TweetTokenizer object at 0x7fd1fc294760>>)),
                ('trans', 'passthrough')])), ('clf', RandomForestClassifier(clas
s_weight='balanced', max_depth=5))]

Best score from GS-CV: 0.723

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.94      0.83      0.88      2374
           1       0.45      0.71      0.55       455

    accuracy                           0.81      2829
   macro avg       0.69      0.77      0.71      2829
weighted avg       0.86      0.81      0.83      2829


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.91      0.81      0.86       594
           1       0.38      0.58      0.46       114

    accuracy                           0.78       708
   macro avg       0.64      0.70      0.66       708
weighted avg       0.82      0.78      0.79       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.7148
         Training Macro Recall = 0.7707
    Training Balanced Accuracy = 0.7707

****************************** Test Scores ******************************
                Test Macro F1 = 0.6574
             Test Macro Recall = 0.6969
        Test Balanced Accuracy = 0.6969

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.05740000000000001
```

```
    Train-Test Macro Recall Diff = -0.07380000000000009
Train-Test Balanced Accuracy Diff = -0.07380000000000009

****************************** Graphs for Test ******************************
```



Logistic Regression

In [372…

```
if rerun_grid:
    # Subbing Logistic Regression for the classifier
    clf_pipe = Pipeline([
        ('prep', prep_pipe),
        ('clf', LogisticRegression(max_iter=500, class_weight='balanced'))
    ])

    # classifier-specific params for LR
    clf_params = {"clf__penalty": ['l1', 'l2', 'elasticnet'],
        "clf__fit_intercept": [True, False],
        "clf__solver": ['newton-cg', 'lbfgs', 'saga'],
        "clf__C": [1, 0.1]}

    # create grid params for LR
    grid_params = [{**common_params, **binary_params, **clf_params},
                   {**common_params, **count_params, **clf_params}]

    model_name = "LR_binary_nostemlem"

    # run gridsearch for LR
    clf_gridsearch_wpipe(clf_pipe, grid_params, X_train, y_train, X_test, y_test
                         class_labels, file_name=model_name, save_path=save_path
                         scoring='recall_macro')
```

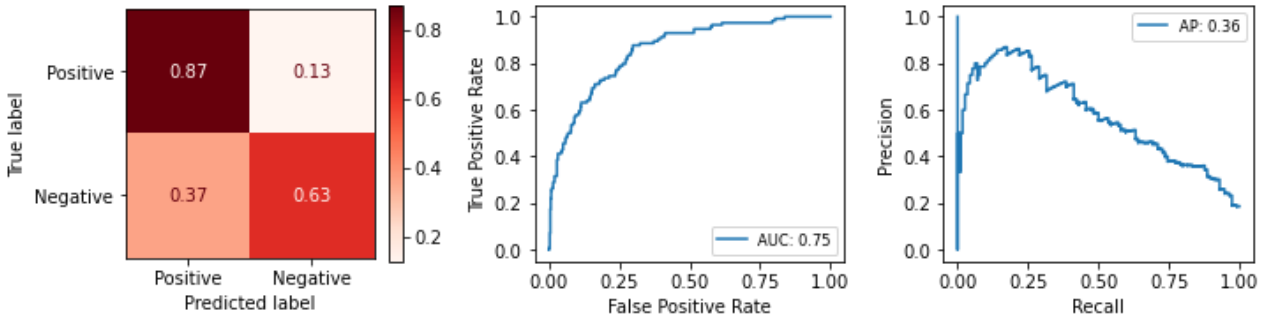
In [373…

```
# load and eval pipeline with best params from colab gridsearch
best_LR_binary_nostemlem_pipe, best_LR_binary_nostemlem_gs = \
load_rebuild_eval_bestpipe('GSObject_LR_binary_nostemlem.joblib.gz',
                           X_train, y_train, X_test, y_test,
                           class_labels, load_path=save_path)
```

```
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator CountVectorizer fro
m version 0.22.2.post1 when using version 0.23.2. This might lead to breaking co
de or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator TfidfTransformer fr
om version 0.22.2.post1 when using version 0.23.2. This might lead to breaking c
ode or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator Pipeline from versi
on 0.22.2.post1 when using version 0.23.2. This might lead to breaking code or i
```

```
nvalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator LogisticRegression
from version 0.22.2.post1 when using version 0.23.2. This might lead to breaking
code or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator GridSearchCV from v
ersion 0.22.2.post1 when using version 0.23.2. This might lead to breaking code
or invalid results. Use at your own risk.
  warnings.warn(
[('prep', Pipeline(steps=[('vect',
                 CountVectorizer(stop_words=['a', 'an', 'and', 'am', 'are',
                                             'as', 'at', 'be', 'by', 'for',
                                             'from', 'if', 'in', 'is', 'into',
                                             'it', "it's", 'its', 'itself',
                                             'of', 'on', 'or', 'than', 'that',
                                             'the', 'to', 'austin', 'sxsw',
                                             '#sxsw', 'link', ...],
                                 tokenizer=<bound method TweetTokenizer.tokenize
of <nltk.tokenize.casual.TweetTokenizer object at 0x7fd1fc34ad60>>)),
                ('trans', TfidfTransformer())])), ('clf', LogisticRegression(C=
1, class_weight='balanced', fit_intercept=False,
                   max_iter=500, solver='newton-cg'))]

Best score from GS-CV: 0.76

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       1.00      0.93      0.96      2374
           1       0.74      0.98      0.84       455

    accuracy                           0.94      2829
   macro avg       0.87      0.96      0.90      2829
weighted avg       0.95      0.94      0.94      2829


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.92      0.87      0.90       594
           1       0.48      0.63      0.55       114

    accuracy                           0.83       708
   macro avg       0.70      0.75      0.72       708
weighted avg       0.85      0.83      0.84       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.902
         Training Macro Recall = 0.9564
   Training Balanced Accuracy = 0.9564

****************************** Test Scores ******************************
                Test Macro F1 = 0.7222
             Test Macro Recall = 0.751
        Test Balanced Accuracy = 0.751

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.17980000000000007
       Train-Test Macro Recall Diff = -0.20540000000000003
Train-Test Balanced Accuracy Diff = -0.20540000000000003
```

```
****************************** Graphs for Test ******************************
```



Looking at the best model from each of these, I'm seeing that they're all pretty overfit to the training data, with a difference of 8-12% on macro F1 and macro recall between train and test on most.

Since I'm optimizing for `recall_macro`, this is balancing the recall for both the positive and negative classes. When I run again using different train-test-splits, I find the balance between recall for positive and negative varies a lot. I've seen as high as 72% for true negative class using these same parameters, although the TTS in this iteration is only 61%.

In [374…
```
# what would be the best recall (optimizing for the negative class only)?
xval = cross_val_score(best_LR_binary_nostemlem_pipe, X_train, y_train,
                       scoring='recall', n_jobs=-1, verbose=True)

print(f"All scores: {xval}")
print(f"Average x-validated score (recall): {np.round(np.mean(xval),3)}")
print()
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
All scores: [0.68131868 0.61538462 0.69230769 0.71428571 0.63736264]
Average x-validated score (recall): 0.668

[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:   42.0s finished
```

Try out some stemming and lemmatization on the best models

I'm not going to gridsearch again with stemming and lemmatization, I will just test the best model params from gridsearching without stemming and lemmatization, and see if it improves performance.

In [375…
```
#nltk.download('wordnet')
```

In [376…
```
stemmer = PorterStemmer()
stemmer.stem('testing')
```

Out[376… `'test'`

In [377…
```
lemmatizer = WordNetLemmatizer()
lemmatizer.lemmatize("testing")
```

Out[377… `'testing'`

In [378…
```
lemmatizer.lemmatize("feet")
```

Out[378…
```
'foot'
```

Hmm, so the lemmatizer is not doing the best job here even with a simple word like `testing`. Stemming did much better with that.

Maybe I'll try passing through lemmatizer first (which will leave words it can't do anything with alone, but change what it can) and then do stemming separately.

In [379…
```
def tokenize_lemma_stem(doc):
    """
    Applies TweetTokenization, then lemmatization and stemming in one shot.

    Uses NLTK, so assumes the appropriate NLTK classes have been imported.

    Uses TweetTokenizer to tokenize documents first, and remove handles.
    Then uses NLTK lemmatization on each token.
    Finally, applies stemming to each token.
    """
    tweettokenizer = TweetTokenizer(preserve_case=False, strip_handles=True)
    lemmatizer = WordNetLemmatizer()
    stemmer = PorterStemmer()

    # tokenize using TweetTokenizer
    tokens = tweettokenizer.tokenize(doc)

    # lemmatize using NLTK
    tokens = [lemmatizer.lemmatize(token) for token in tokens]

    # stem using NLTK
    tokens = [stemmer.stem(token, ) for token in tokens]

    return tokens
```

Let's test it out!

In [380…
```
tokenize_lemma_stem("This isn't what I was hoping for. I'm testing. Feet and han
```

Out[380…
```
['thi',
 "isn't",
 'what',
 'i',
 'wa',
 'hope',
 'for',
 '.',
 "i'm",
 'test',
 '.',
 'foot',
 'and',
 'hand',
 '.',
 'what',
 'do',
 'you',
 'think',
 '?',
```

```
'think',
'?']
```

Multinomial Bayes

In [381…

```
from sklearn.base import clone

# copy the best MNB pipeline/model from the gridsearch (deep copy)
MNB_binary_withstemlem_pipe = clone(best_MNB_binary_nostemlem_pipe)

# set tokenizer in the copy to use the function with stemming and lemmatization
MNB_binary_withstemlem_pipe.set_params(prep__vect__tokenizer=tokenize_lemma_stem

# check params
MNB_binary_withstemlem_pipe
```

Out[381…

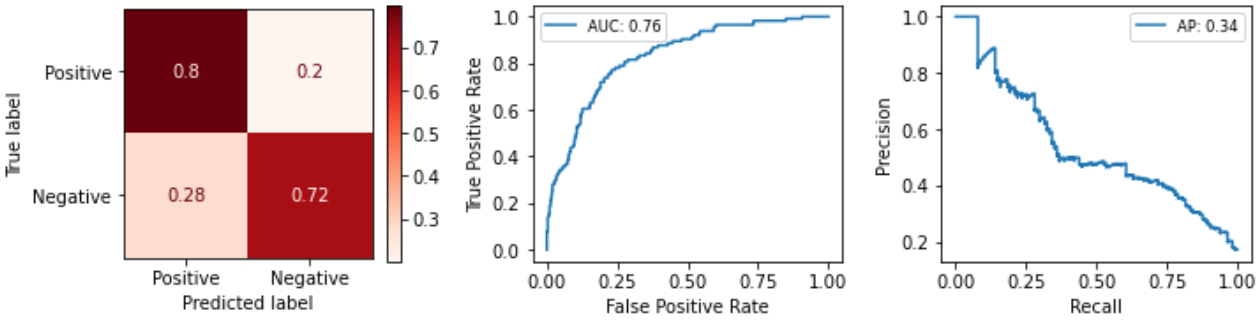
```
Pipeline(steps=[('prep',
                 Pipeline(steps=[('vect',
                                  CountVectorizer(binary=True,
                                                  max_features=1000,
                                                  stop_words=['google', 'apple',
                                                              'ipad', 'iphone',
                                                              'android',
                                                              'ipad2',
                                                              '#google',
                                                              '#apple', '#ipad',
                                                              '#iphone',
                                                              '#android',
                                                              '#ipad2', 'a',
                                                              'an', 'and', 'am',
                                                              'are', 'as', 'at',
                                                              'be', 'by', 'for',
                                                              'from', 'if',
                                                              'in', 'is',
                                                              'into', 'it',
                                                              "it's", 'its',
...],
                                                  tokenizer=<function tokenize_l
emma_stem at 0x7fd209e47550>)),
                                 ('trans', 'passthrough')])),
                ('clf', MultinomialNB(fit_prior=False))])
```

In [382…

```
# get cross_val_scores for best MNB model with stemming/lemmatization
xval = cross_val_score(MNB_binary_withstemlem_pipe, X_train, y_train,
                       scoring='recall_macro', n_jobs=-1, verbose=True)

print(f"All scores: {xval}")
print(f"Average x-validated score (recall): {np.round(np.mean(xval),3)}")
print()

# fit model to generate report
MNB_binary_withstemlem_pipe.fit(X_train, y_train)

eval_clf_model(MNB_binary_withstemlem_pipe, X_test, y_test, X_train, y_train,
               labels=class_labels)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:   11.3s finished
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/feature_extraction/text.py:383: UserWarning: Your stop_words may be incons
istent with your preprocessing. Tokenizing the stop words generated tokens ['#ap
pl', '#googl', '#iphon', '2', 'appl', 'googl', 'iphon', "it'"] not in stop_word
```

```
s.
  warnings.warn('Your stop_words may be inconsistent with '
All scores: [0.74335454 0.70663968 0.75573164 0.71470214 0.77258079]
Average x-validated score (recall): 0.739

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.97      0.83      0.90      2374
           1       0.50      0.88      0.64       455

    accuracy                           0.84      2829
   macro avg       0.74      0.85      0.77      2829
weighted avg       0.90      0.84      0.85      2829


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.94      0.80      0.86       594
           1       0.41      0.72      0.52       114

    accuracy                           0.79       708
   macro avg       0.67      0.76      0.69       708
weighted avg       0.85      0.79      0.81       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.766
         Training Macro Recall = 0.854
    Training Balanced Accuracy = 0.854

****************************** Test Scores ******************************
                Test Macro F1 = 0.6917
             Test Macro Recall = 0.7595
        Test Balanced Accuracy = 0.7595

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.07430000000000003
       Train-Test Macro Recall Diff = -0.09450000000000003
Train-Test Balanced Accuracy Diff = -0.09450000000000003

****************************** Graphs for Test ******************************
```



Not much different from the cross-validated recall_macro score without stemming and lemmatization. In fact this version is slightly worse.

I ended up running a gridsearch will all the params in colab just to see if a different combination of other parameters could result in better performance using stemming/lemmatization, but it was about the same if not slightly worse.

Logistic Regression

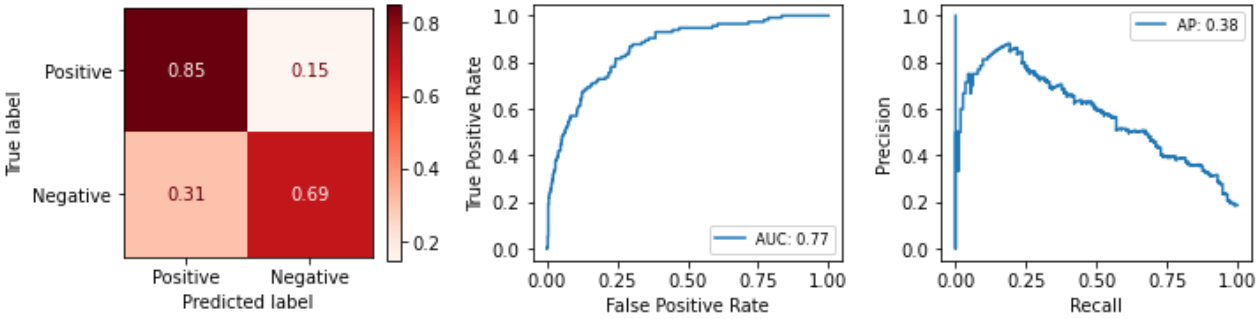
Best params without stemming/lemmatization, with stemming/lemmatization applied.

In [383…

```
# copy the best MNB pipeline/model from the gridsearch (deep copy)
LR_binary_withstemlem_pipe = clone(best_LR_binary_nostemlem_pipe)

# set tokenizer in the copy to use the function with stemming and lemmatization
LR_binary_withstemlem_pipe.set_params(prep__vect__tokenizer=tokenize_lemma_stem)

# get cross_val_scores for best MNB model with stemming/lemmatization
xval = cross_val_score(LR_binary_withstemlem_pipe, X_train, y_train,
                       scoring='recall_macro', n_jobs=-1, verbose=True)

print(f"All scores: {xval}")
print(f"Average x-validated score (recall): {np.round(np.mean(xval),3)}")
print()

# fit model to generate report
LR_binary_withstemlem_pipe.fit(X_train, y_train)

eval_clf_model(LR_binary_withstemlem_pipe, X_test, y_test, X_train, y_train,
               labels=class_labels)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    5.2s finished
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/feature_extraction/text.py:383: UserWarning: Your stop_words may be incons
istent with your preprocessing. Tokenizing the stop words generated tokens ["i
t'"] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
All scores: [0.78510121 0.73226142 0.75049161 0.77773279 0.77464413]
Average x-validated score (recall): 0.764

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       1.00      0.92      0.96      2374
           1       0.71      0.98      0.82       455

    accuracy                           0.93      2829
   macro avg       0.85      0.95      0.89      2829
weighted avg       0.95      0.93      0.94      2829


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.94      0.85      0.89       594
           1       0.47      0.69      0.56       114

    accuracy                           0.83       708
   macro avg       0.70      0.77      0.73       708
weighted avg       0.86      0.83      0.84       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.8911
         Training Macro Recall = 0.9502
   Training Balanced Accuracy = 0.9502

****************************** Test Scores ******************************
```

```
                Test Macro F1 = 0.727
             Test Macro Recall = 0.7724
         Test Balanced Accuracy = 0.7724

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.16410000000000002
       Train-Test Macro Recall Diff = -0.17780000000000007
Train-Test Balanced Accuracy Diff = -0.17780000000000007

****************************** Graphs for Test ******************************
```



Gridsearch for Logistic Regression

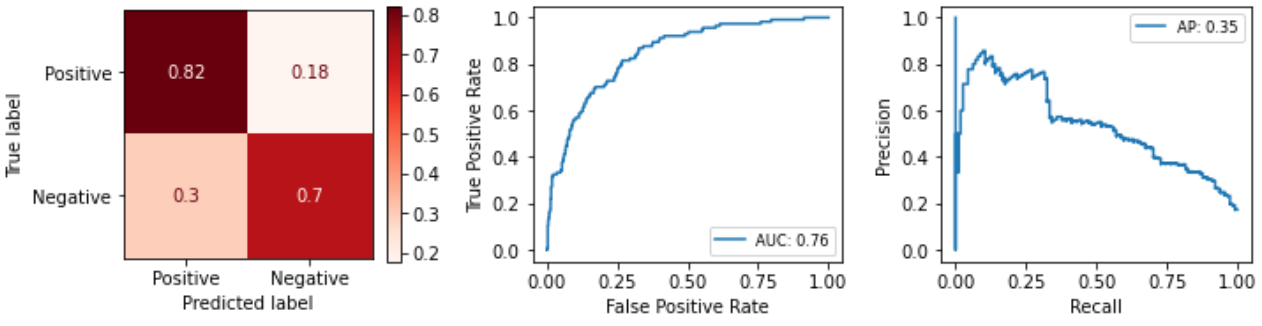
Gridsearch was run in colab, and got different best params with stem/lemma than without.

In [384…

```
# load and eval pipeline with best params from colab gridsearch
best_LR_binary_withstemlem_pipe, best_LR_binary_withstemlem_gs = \
load_rebuild_eval_bestpipe('GSObject_LR_binary_withstemlem.joblib.gz',
                           X_train, y_train, X_test, y_test,
                           class_labels, load_path=save_path)
```

```
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator CountVectorizer fro
m version 0.22.2.post1 when using version 0.23.2. This might lead to breaking co
de or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator TfidfTransformer fr
om version 0.22.2.post1 when using version 0.23.2. This might lead to breaking c
ode or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator Pipeline from versi
on 0.22.2.post1 when using version 0.23.2. This might lead to breaking code or i
nvalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator LogisticRegression
from version 0.22.2.post1 when using version 0.23.2. This might lead to breaking
code or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/base.py:329: UserWarning: Trying to unpickle estimator GridSearchCV from v
ersion 0.22.2.post1 when using version 0.23.2. This might lead to breaking code
or invalid results. Use at your own risk.
  warnings.warn(
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/feature_extraction/text.py:383: UserWarning: Your stop_words may be incons
istent with your preprocessing. Tokenizing the stop words generated tokens ["i
t'"] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
[('prep', Pipeline(steps=[('vect',
```

```
                CountVectorizer(max_features=1000,
                                stop_words=['a', 'an', 'and', 'am', 'are',
                                            'as', 'at', 'be', 'by', 'for',
                                            'from', 'if', 'in', 'is', 'into',
                                            'it', "it's", 'its', 'itself',
                                            'of', 'on', 'or', 'than', 'that',
                                            'the', 'to', 'austin', 'sxsw',
                                            '#sxsw', 'link', ...],
                                tokenizer=<function tokenize_lemma_stem at 0x7f
d209e47550>)),
               ('trans', TfidfTransformer())])), ('clf', LogisticRegression(C=
1, class_weight='balanced', max_iter=500, solver='saga'))]

Best score from GS-CV: 0.761

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.99      0.88      0.93      2374
           1       0.59      0.93      0.72       455

    accuracy                           0.89      2829
   macro avg       0.79      0.91      0.83      2829
weighted avg       0.92      0.89      0.90      2829


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.93      0.82      0.87       594
           1       0.43      0.70      0.53       114

    accuracy                           0.80       708
   macro avg       0.68      0.76      0.70       708
weighted avg       0.85      0.80      0.82       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.8263
         Training Macro Recall = 0.9053
   Training Balanced Accuracy = 0.9053

****************************** Test Scores ******************************
               Test Macro F1 = 0.7039
            Test Macro Recall = 0.7617
       Test Balanced Accuracy = 0.7617

****************************** Differences ******************************
         Train-Test Macro F1 Diff = -0.12240000000000006
     Train-Test Macro Recall Diff = -0.14359999999999995
Train-Test Balanced Accuracy Diff = -0.14359999999999995

****************************** Graphs for Test ******************************
```



Although gridsearch specifically including the stem/lemma preprocessing yielded different best params than without stem/lemma, the overall recall_macro score on cross validation is not significantly different, knowing that scores at this point are varying quite a bit depending on the specific makeup of train-test-split.

Modeling Multi-class

Initially, I built models to parse out the positive or negative product sentiment, to understand what people said that contributed to each one. However, in order to use that model on other/future datasets, we would first need to separate out the tweets that had NO emotion towards products or brands, versus those that had positive or negative emotion.

I'm going to see if I can build a multi-class model that can perform as well at classifying positive and negative as my binary model (as well as classifying no emotion). I suspect this may be difficult to achieve.

The other option is to try to construct a multi-class model that will predict 'no emotion' very accurately, which would allow me to filter those out. Then the remaining 'some emotion' tweets could be fed into the binary model as a second step.

Preprocessing for multi-class modeling

In [385…
```
df['emotion'].value_counts(1)
```

Out[385…
```
No emotion toward brand or product    0.592609
Positive emotion                      0.327413
Negative emotion                      0.062769
I can't tell                          0.017209
Name: emotion, dtype: float64
```

In [386…
```
Xm = df.loc[df['emotion'].isin(['Positive emotion', 'Negative emotion',
                                'No emotion toward brand or product']),
            'cleaned']
ym = df.loc[df['emotion'].isin(['Positive emotion', 'Negative emotion',
                                'No emotion toward brand or product']),
            'emotion']

print(len(Xm))
print(len(ym))
```

```
8909
8909
```

In [387…
```
ym = ym.map(lambda x: 0 if x=="Negative emotion" else \
            (1 if x=="No emotion toward brand or product" else 2))
ym.value_counts()
```

Out[387…
```
1    5372
2    2968
0     569
Name: emotion, dtype: int64
```

In [388…
```
# 0 is negative, 1 is none, 2 is positive
mclass_labels = ['Negative', 'None', 'Positive']
```

In [389…
```
# train test split
Xm_train, Xm_test, ym_train, ym_test = train_test_split(Xm, ym, test_size=0.2,
                                                        stratify=ym)
print(len(Xm_train))
print(len(ym_train))
print(len(Xm_test))
print(len(ym_test))
```

```
7127
7127
1782
1782
```

In [390…
```
ym_train.value_counts(normalize=True)
```

Out[390…
```
1    0.603059
2    0.333099
0    0.063842
Name: emotion, dtype: float64
```

In [391…
```
ym_test.value_counts(normalize=True)
```

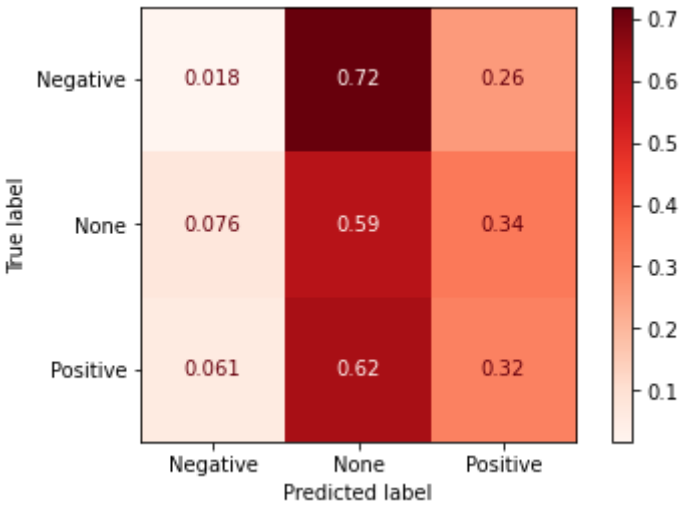
Out[391…
```
1    0.602694
2    0.333333
0    0.063973
Name: emotion, dtype: float64
```

Dummy Classifier for Multiclass

In [392…
```
prep_pipe = Pipeline([
    ('vect', CountVectorizer(tokenizer=tweettokenizer.tokenize)),
    ('trans', TfidfTransformer())
])

# modling pipeline with preprocessing and model built in
clf_pipe = Pipeline([
    ('prep', prep_pipe),
    ('clf', DummyClassifier(strategy='stratified'))
])

clf_pipe.fit(Xm_train, ym_train)

eval_clf_model(clf_pipe, Xm_test, ym_test, Xm_train, ym_train,
               labels=mclass_labels, normalize_cm='true')
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.08      0.08      0.08       455
           1       0.60      0.61      0.60      4298
           2       0.33      0.32      0.33      2374

    accuracy                           0.48      7127
   macro avg       0.34      0.34      0.34      7127
weighted avg       0.48      0.48      0.48      7127


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.06      0.06      0.06       114
```

```
           1       0.60      0.59      0.59      1074
           2       0.31      0.31      0.31       594

    accuracy                           0.47      1782
   macro avg       0.32      0.32      0.32      1782
weighted avg       0.47      0.47      0.47      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.3375
         Training Macro Recall = 0.3376
    Training Balanced Accuracy = 0.3376

****************************** Test Scores ******************************
                Test Macro F1 = 0.3235
             Test Macro Recall = 0.3234
        Test Balanced Accuracy = 0.3234

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.014000000000000012
       Train-Test Macro Recall Diff = -0.01419999999999999
Train-Test Balanced Accuracy Diff = -0.01419999999999999

****************************** Graphs for Test ******************************
```



Baseline Multinomial Bayes Models

In [393…

```
clf_pipe = Pipeline([
    ('prep', prep_pipe),
    ('clf', MultinomialNB())
])
```

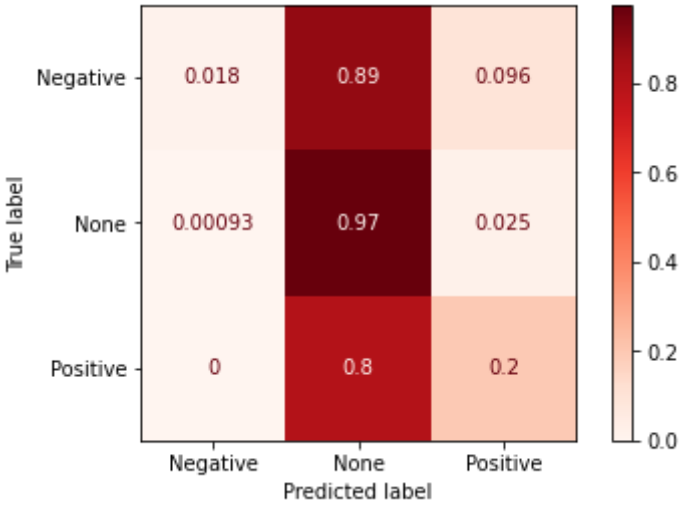
Tfidf Standardized Doc Term Matrix

In [394…

```
clf_pipe.fit(Xm_train, ym_train)

eval_clf_model(clf_pipe, Xm_test, ym_test, Xm_train, ym_train,
               labels=mclass_labels)
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       1.00      0.01      0.01       455
           1       0.70      0.99      0.82      4298
```

```
              2       0.91      0.40      0.55      2374

    accuracy                           0.73      7127
   macro avg       0.87      0.47      0.46      7127
weighted avg       0.79      0.73      0.68      7127


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.67      0.02      0.03       114
           1       0.64      0.97      0.78      1074
           2       0.75      0.20      0.31       594

    accuracy                           0.65      1782
   macro avg       0.69      0.40      0.37      1782
weighted avg       0.68      0.65      0.57      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.4622
         Training Macro Recall = 0.465
    Training Balanced Accuracy = 0.465

****************************** Test Scores ******************************
                Test Macro F1 = 0.374
             Test Macro Recall = 0.3961
        Test Balanced Accuracy = 0.3961

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.0882
       Train-Test Macro Recall Diff = -0.06890000000000002
Train-Test Balanced Accuracy Diff = -0.06890000000000002

****************************** Graphs for Test ******************************
```



Well that's definitely worse than the dummy; it just pretty much always predicts the 'no emotion' class.

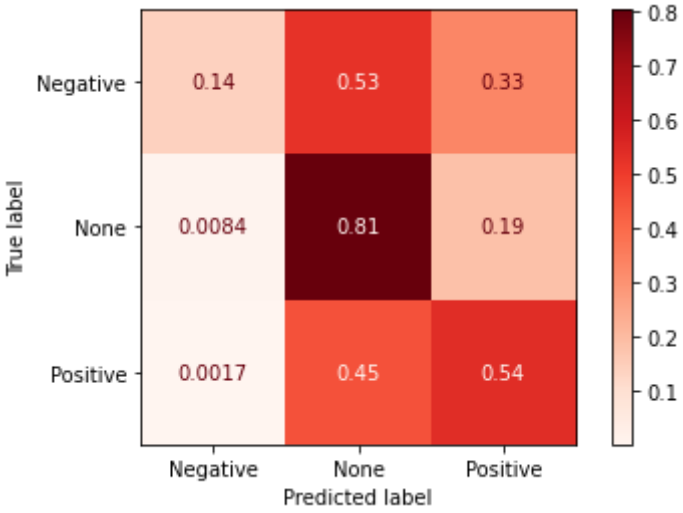
Counted Doc Term Matrix

In [395…

```
# Since my pipeline includes the Tfidf transformer, I'm explicitly setting
# that pipeline step to be passthrough here to remove it
clf_pipe.set_params(prep__vect__binary=False)
clf_pipe.set_params(prep__trans='passthrough')
```

```
clf_pipe.fit(Xm_train, ym_train)

eval_clf_model(clf_pipe, Xm_test, ym_test, Xm_train, ym_train,
               labels=mclass_labels)
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.83      0.41      0.55       455
           1       0.84      0.89      0.86      4298
           2       0.75      0.75      0.75      2374

    accuracy                           0.81      7127
   macro avg       0.81      0.68      0.72      7127
weighted avg       0.81      0.81      0.80      7127


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.62      0.14      0.23       114
           1       0.72      0.81      0.76      1074
           2       0.58      0.54      0.56       594

    accuracy                           0.68      1782
   macro avg       0.64      0.50      0.52      1782
weighted avg       0.67      0.68      0.66      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.7198
         Training Macro Recall = 0.681
   Training Balanced Accuracy = 0.681

****************************** Test Scores ******************************
                Test Macro F1 = 0.5168
             Test Macro Recall = 0.4965
       Test Balanced Accuracy = 0.4965

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.20299999999999996
      Train-Test Macro Recall Diff = -0.18450000000000005
Train-Test Balanced Accuracy Diff = -0.18450000000000005

****************************** Graphs for Test ******************************
```



Not bad on the None class, but we're only classifying 19% of Negatives correctly and 56% of

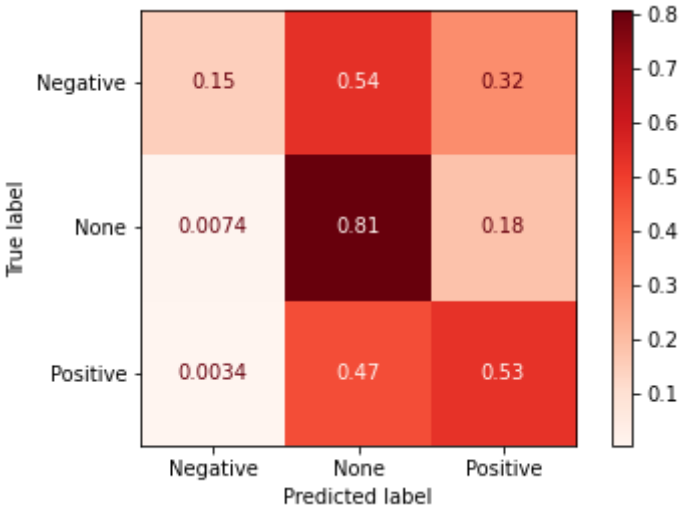
positives correctly.

Binary Doc Term Matrix

In [396…

```
# Let's try the binary representation instead of raw counts
clf_pipe.set_params(prep__vect__binary=True)
clf_pipe.set_params(prep__trans='passthrough')

clf_pipe.fit(Xm_train, ym_train)

eval_clf_model(clf_pipe, Xm_test, ym_test, Xm_train, ym_train,
               labels=mclass_labels)
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.86      0.41      0.55       455
           1       0.84      0.90      0.87      4298
           2       0.77      0.75      0.76      2374

    accuracy                           0.82      7127
   macro avg       0.82      0.68      0.73      7127
weighted avg       0.82      0.82      0.81      7127


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.63      0.15      0.24       114
           1       0.72      0.81      0.76      1074
           2       0.57      0.53      0.55       594

    accuracy                           0.67      1782
   macro avg       0.64      0.50      0.52      1782
weighted avg       0.67      0.67      0.66      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.7268
         Training Macro Recall = 0.6847
    Training Balanced Accuracy = 0.6847

****************************** Test Scores ******************************
                Test Macro F1 = 0.5179
             Test Macro Recall = 0.4959
        Test Balanced Accuracy = 0.4959

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.20889999999999997
       Train-Test Macro Recall Diff = -0.18879999999999997
Train-Test Balanced Accuracy Diff = -0.18879999999999997

****************************** Graphs for Test ******************************
```



These are all pretty bad. They predict the 'no emotion' class fairly accurately, and where they misclassify they tend to skew towards positive. They're all quite bad at classifying negative emotions.

GridSearchCVs for best params for various models

In [397…

```
# Params that will apply to the vectorizer regardless of how the
# document term matrix is constructed
common_params = {"prep__vect__stop_words": [custom_stopwords + punc_custom,
            nltk_stopwords + punc_custom,
            product_stopwords + custom_stopwords + punc_custom,
            punc_custom + product_stopwords + nltk_stopwords],
    "prep__vect__ngram_range": [(1,1), (1,2), (2,2)],
    "prep__vect__max_features": [None, 1000]}

# Params for the binary frequency, where we don't want to bother with a Tfidf
binary_params = {
    "prep__vect__binary": [True],
    "prep__trans": ['passthrough']}

# params for non-binary frequency, where we want to test both with and without
# Tfidf
count_params = {
    "prep__vect__binary": [False],
    "prep__trans": [TfidfTransformer(), 'passthrough']}
```

Multinomial Bayes No stemming

In [398…

```
if rerun_grid:
    # pre-processing pipeline with NO stemming or lemmatization
    prep_pipe = Pipeline([
        ('vect', CountVectorizer(tokenizer=tweettokenizer.tokenize)),
        ('trans', TfidfTransformer())
    ])

    # main pipeline with clf
    clf_pipe = Pipeline([
        ('prep', prep_pipe),
        ('clf', MultinomialNB())
    ])
```

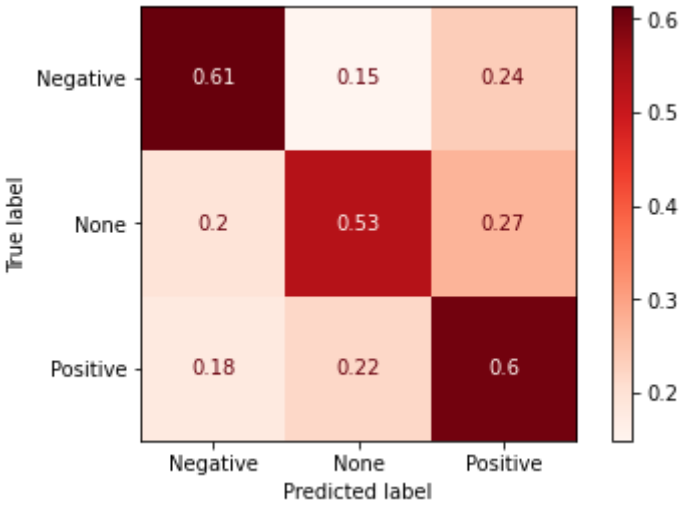
```
    # classifier-specific params for MNB
    clf_params = {"clf__fit_prior": [True, False]}

    # create grid params for MNB
    grid_params = [{**common_params, **binary_params, **clf_params},
                   {**common_params, **count_params, **clf_params}]

    model_name = "MNB_multiclass_nostemlem"

    # run gridsearch for MNB
    clf_gridsearch_wpipe(clf_pipe, grid_params, Xm_train, ym_train, Xm_test, ym_
                         mclass_labels, file_name=model_name, save_path=save_pat
                         scoring='recall_macro')
```

In [399…

```
# load and eval pipeline with best params from gridsearch
best_MNB_multiclass_nostemlem_pipe, best_MNB_multiclass_gs = \
load_rebuild_eval_bestpipe('GSObject_MNB_multiclass_nostemlem.joblib.gz',
                           Xm_train, ym_train, Xm_test, ym_test,
                           mclass_labels, load_path=save_path)
```

```
[('prep', Pipeline(steps=[('vect',
                 CountVectorizer(max_features=1000,
                                 stop_words=['a', 'an', 'and', 'am', 'are',
                                             'as', 'at', 'be', 'by', 'for',
                                             'from', 'if', 'in', 'is', 'into',
                                             'it', "it's", 'its', 'itself',
                                             'of', 'on', 'or', 'than', 'that',
                                             'the', 'to', 'austin', 'sxsw',
                                             '#sxsw', 'link', ...],
                                 tokenizer=<bound method TweetTokenizer.tokenize
of <nltk.tokenize.casual.TweetTokenizer object at 0x7fd1e00cf670>>)),
                ('trans', 'passthrough')])), ('clf', MultinomialNB(fit_prior=Fal
se))]

Best score from GS-CV: 0.565

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.23      0.78      0.36       455
           1       0.82      0.55      0.66      4298
           2       0.57      0.65      0.61      2374

    accuracy                           0.60      7127
   macro avg       0.54      0.66      0.54      7127
weighted avg       0.70      0.60      0.62      7127


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.18      0.61      0.28       114
           1       0.79      0.53      0.64      1074
           2       0.53      0.60      0.56       594

    accuracy                           0.56      1782
   macro avg       0.50      0.58      0.49      1782
weighted avg       0.67      0.56      0.59      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.5402
         Training Macro Recall = 0.6602
```

```
   Training Balanced Accuracy = 0.6602

****************************** Test Scores ******************************
                Test Macro F1 = 0.4926
             Test Macro Recall = 0.5822
       Test Balanced Accuracy = 0.5822

***************************** Differences *****************************
         Train-Test Macro F1 Diff = -0.04760000000000003
      Train-Test Macro Recall Diff = -0.07799999999999996
Train-Test Balanced Accuracy Diff = -0.07799999999999996

***************************** Graphs for Test *****************************
```



Not quite as good at predicting negative and positive emotions, based on comparing True Negatives and Positives. But not too bad.

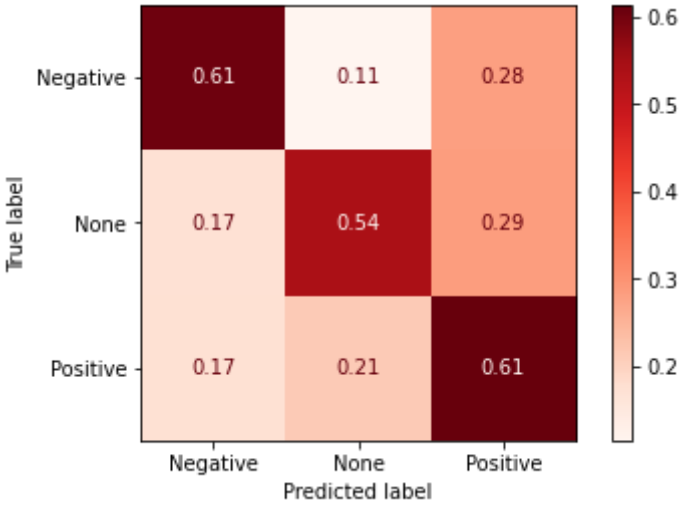
Multinomial Bayes with Stemming and Lemmatization

In [400…

```
# copy the best MNB pipeline/model from the gridsearch (deep copy)
MNB_multiclass_withstemlem_pipe = clone(best_MNB_multiclass_nostemlem_pipe)

# set tokenizer in the copy to use the function with stemming and lemmatization
MNB_multiclass_withstemlem_pipe.set_params(prep__vect__tokenizer=tokenize_lemma_

# get cross_val_scores for best MNB model with stemming/lemmatization
xval = cross_val_score(MNB_multiclass_withstemlem_pipe, Xm_train, ym_train,
                       scoring='recall_macro', n_jobs=-1, verbose=True)

print(f"All scores: {xval}")
print(f"Average x-validated score (recall): {np.round(np.mean(xval),3)}")
print()

# fit model to generate report
MNB_multiclass_withstemlem_pipe.fit(Xm_train, ym_train)

eval_clf_model(MNB_multiclass_withstemlem_pipe, Xm_test, ym_test, Xm_train,
               ym_train, labels=mclass_labels)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:   18.5s finished
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/feature_extraction/text.py:383: UserWarning: Your stop_words may be incons
istent with your preprocessing. Tokenizing the stop words generated tokens ["i
```

```
t'"] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
All scores: [0.59184962 0.56966755 0.59201788 0.57255589 0.57272122]
Average x-validated score (recall): 0.58

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.25      0.79      0.38       455
           1       0.83      0.56      0.67      4298
           2       0.57      0.67      0.61      2374

    accuracy                           0.61      7127
   macro avg       0.55      0.67      0.55      7127
weighted avg       0.71      0.61      0.63      7127


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.19      0.61      0.29       114
           1       0.81      0.54      0.65      1074
           2       0.52      0.61      0.56       594

    accuracy                           0.57      1782
   macro avg       0.51      0.59      0.50      1782
weighted avg       0.67      0.57      0.60      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.5534
         Training Macro Recall = 0.6731
    Training Balanced Accuracy = 0.6731

****************************** Test Scores ******************************
                Test Macro F1 = 0.5013
             Test Macro Recall = 0.5872
        Test Balanced Accuracy = 0.5872

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.052100000000000035
       Train-Test Macro Recall Diff = -0.08589999999999998
Train-Test Balanced Accuracy Diff = -0.08589999999999998

****************************** Graphs for Test ******************************
```



Stemming/lemmatization doesn't appear to help performance much.

Logistic Regression No Stemming

In [401…

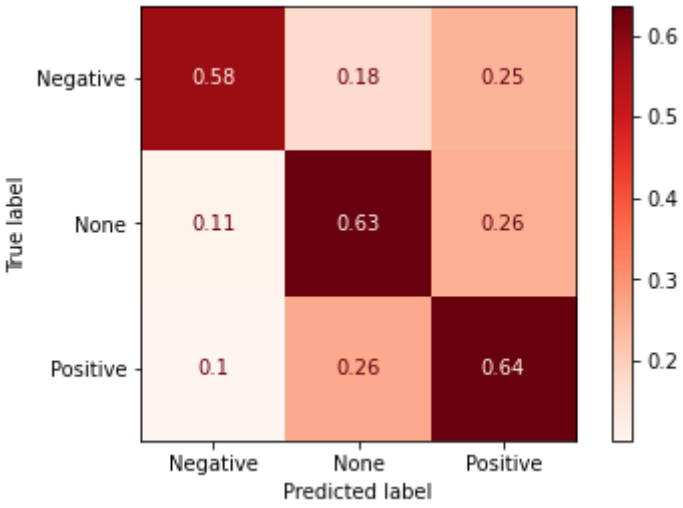
```
if rerun_grid:
    clf_pipe = Pipeline([
        ('prep', prep_pipe),
        ('clf', LogisticRegression(max_iter=500, class_weight='balanced'))
    ])

    # create a dict of params for LR
    clf_params = {"clf__penalty": ['l1', 'l2', 'elasticnet'],
        "clf__fit_intercept": [True, False],
        "clf__solver": ['newton-cg', 'lbfgs', 'saga'],
        "clf__C": [1, 0.1]}

    # create grid params for LR
    grid_params = [{**common_params, **binary_params, **clf_params},
                   {**common_params, **count_params, **clf_params}]

    model_name = "LR_multiclass_nostemlem"

    # run gridsearch for LR
    clf_gridsearch_wpipe(clf_pipe, grid_params, Xm_train, ym_train, Xm_test, ym_
                         mclass_labels, file_name=model_name, save_path=save_pat
                         scoring='recall_macro')
```

In [402…

```
# load and eval pipeline with best params from gridsearch
best_LR_multiclass_nostemlem_pipe, best_LR_multiclass_gs = \
load_rebuild_eval_bestpipe('GSObject_LR_multiclass_nostemlem.joblib.gz',
                           Xm_train, ym_train, Xm_test, ym_test,
                           mclass_labels, load_path=save_path)
```

```
[('prep', Pipeline(steps=[('vect',
                 CountVectorizer(ngram_range=(1, 2),
                                 stop_words=['a', 'an', 'and', 'am', 'are',
                                             'as', 'at', 'be', 'by', 'for',
                                             'from', 'if', 'in', 'is', 'into',
                                             'it', "it's", 'its', 'itself',
                                             'of', 'on', 'or', 'than', 'that',
                                             'the', 'to', 'austin', 'sxsw',
                                             '#sxsw', 'link', ...],
                                 tokenizer=<bound method TweetTokenizer.tokenize
of <nltk.tokenize.casual.TweetTokenizer object at 0x7fd1fc8b70a0>>)),
                ('trans', TfidfTransformer())])), ('clf', LogisticRegression(C=
0.1, class_weight='balanced', max_iter=500,
                   solver='newton-cg'))]

Best score from GS-CV: 0.602

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.48      0.95      0.63       455
           1       0.88      0.75      0.81      4298
           2       0.71      0.77      0.74      2374

    accuracy                           0.77      7127
   macro avg       0.69      0.82      0.73      7127
weighted avg       0.80      0.77      0.78      7127


****************************** Test Data ******************************
              precision    recall  f1-score   support
```

```
              0       0.27      0.58      0.37       114
              1       0.79      0.63      0.71      1074
              2       0.56      0.64      0.59       594

    accuracy                            0.63      1782
   macro avg       0.54      0.62      0.56      1782
weighted avg       0.68      0.63      0.65      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.7279
         Training Macro Recall = 0.8234
   Training Balanced Accuracy = 0.8234

****************************** Test Scores ******************************
                Test Macro F1 = 0.5555
             Test Macro Recall = 0.6165
       Test Balanced Accuracy = 0.6165

****************************** Differences ******************************
         Train-Test Macro F1 Diff = -0.1724
     Train-Test Macro Recall Diff = -0.20689999999999997
Train-Test Balanced Accuracy Diff = -0.20689999999999997

****************************** Graphs for Test ******************************
```



Logistic Regression with Stemming and Lemmatization

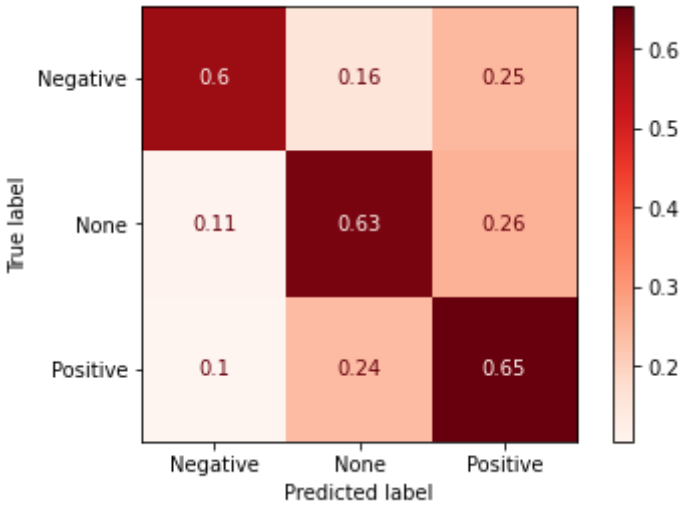
Let's try the best params from gridsearch with stemming and lemmatization and see if that helps?

In [403…

```
# copy the best LR pipeline/model from the gridsearch (deep copy)
LR_multiclass_withstemlem_pipe = clone(best_LR_multiclass_nostemlem_pipe)

# set tokenizer in the copy to use the function with stemming and lemmatization
LR_multiclass_withstemlem_pipe.set_params(prep__vect__tokenizer=tokenize_lemma_s

# get cross_val_scores for best MNB model with stemming/lemmatization
xval = cross_val_score(LR_multiclass_withstemlem_pipe, Xm_train, ym_train,
                       scoring='recall_macro', n_jobs=-1, verbose=True)

print(f"All scores: {xval}")
print(f"Average x-validated score (recall): {np.round(np.mean(xval),3)}")
print()
```

```
# fit model to generate report
LR_multiclass_withstemlem_pipe.fit(Xm_train, ym_train)

eval_clf_model(LR_multiclass_withstemlem_pipe, Xm_test, ym_test, Xm_train,
               ym_train, labels=mclass_labels)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:   24.2s finished
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/feature_extraction/text.py:383: UserWarning: Your stop_words may be incons
istent with your preprocessing. Tokenizing the stop words generated tokens ["i
t'"] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
All scores: [0.61658561 0.62041257 0.60575729 0.59110231 0.60246827]
Average x-validated score (recall): 0.607

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.46      0.95      0.62       455
           1       0.88      0.74      0.80      4298
           2       0.70      0.76      0.73      2374

    accuracy                           0.76      7127
   macro avg       0.68      0.82      0.72      7127
weighted avg       0.79      0.76      0.77      7127


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.28      0.60      0.38       114
           1       0.81      0.63      0.71      1074
           2       0.56      0.65      0.60       594

    accuracy                           0.64      1782
   macro avg       0.55      0.63      0.56      1782
weighted avg       0.69      0.64      0.65      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.7177
         Training Macro Recall = 0.8157
   Training Balanced Accuracy = 0.8157

****************************** Test Scores ******************************
               Test Macro F1 = 0.564
            Test Macro Recall = 0.6285
       Test Balanced Accuracy = 0.6285

****************************** Differences ******************************
         Train-Test Macro F1 Diff = -0.15370000000000006
      Train-Test Macro Recall Diff = -0.18720000000000003
Train-Test Balanced Accuracy Diff = -0.18720000000000003

****************************** Graphs for Test ******************************
```



Imbalanced Classes - Oversampling

Seeing how overfit pretty much all of my models are, I looked more into the problem. I've been relying on the `class_weight='balanced'` parameter in my logistic regression models, but it hasn't appeared to make much of a difference, and the models are consistently overfitting on train regardless of how I do the trains-test-split.

I realized after doing some research that the issue may very well be related to the ngrams the models are focusing on being rare enough that they may occur only in train and not in test. The vocabulary used to predict test is based on what it saw in train, so if a lot of the most important ngrams occurred only in test and didn't appear in test, we would see consistently poor results in test.

However, the only way to address that particular problem would be to perform oversampling BEFORE train-test-split, which would not be appropriate since we could potentially include a copy of the same tweet in both train and test. Without getting into more complicated models, I don't think there is a way to address the overfitting.

I'm going to use Random Oversampling on the training set anyway, and see whether that improves performance, even if I don't think it will reduce overfitting.

Preprocessing for binary ROS

In [404…
```
from imblearn.over_sampling import RandomOverSampler
```

In [405…
```
# train test split
X2_train, X2_test, y2_train, y2_test = train_test_split(X, y, test_size=0.2,
                                                        stratify=y)

print(len(X2_train))
print(len(y2_train))
print(len(X2_test))
print(len(y2_test))
```

```
2829
2829
```

```
708
708
```

In [406…
```
X2_train = pd.DataFrame(X2_train, columns=['cleaned'])
```

In [407…
```
ros = RandomOverSampler()
X2ros_train, y2ros_train = ros.fit_resample(X2_train,
                                            y2_train)
```

In [408…
```
y2ros_train.value_counts()
```

Out[408…
```
1    2374
0    2374
Name: emotion, dtype: int64
```

In [409…
```
print(len(X2ros_train))
print(len(y2ros_train))
```

```
4748
4748
```

In [410…
```
print(X2ros_train.shape)
print(y2ros_train.shape)
```

```
(4748, 1)
(4748,)
```

In [411…
```
X2ros_train = pd.Series(data=X2ros_train['cleaned'])
X2ros_train.shape
```

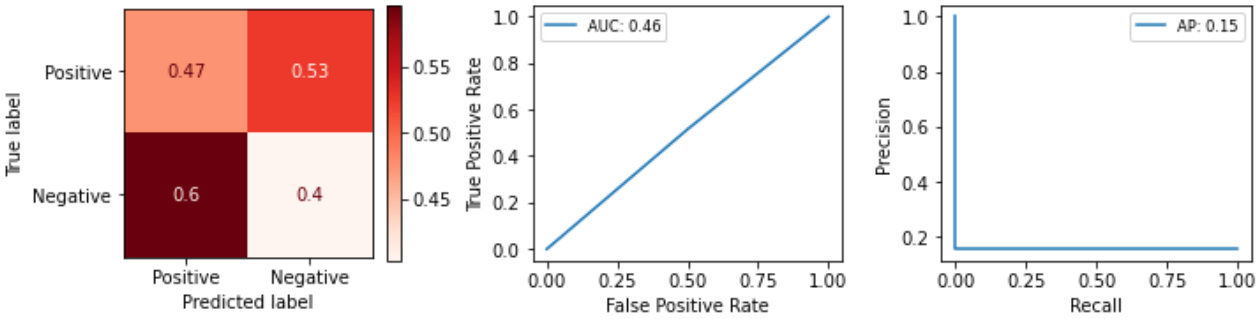
Out[411…
```
(4748,)
```

ROS - Dummy Classifier

In [412…
```
# pre-processing pipeline to transform into vectors
prep_pipe = Pipeline([
    ('vect', CountVectorizer(tokenizer=tweettokenizer.tokenize)),
    ('trans', TfidfTransformer())
])

# Let's make a baseline classifier using the dummy model
clf_pipe = Pipeline([
    ('prep', prep_pipe),
    ('clf', DummyClassifier(strategy='stratified'))
])

clf_pipe.fit(X2ros_train, y2ros_train)

eval_clf_model(clf_pipe, X2_test, y2_test, X2ros_train, y2ros_train,
               labels=class_labels)
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.49      0.48      0.49      2374
           1       0.49      0.50      0.50      2374

    accuracy                           0.49      4748
   macro avg       0.49      0.49      0.49      4748
weighted avg       0.49      0.49      0.49      4748
```

```
****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.82      0.47      0.60       594
           1       0.14      0.46      0.22       114

    accuracy                           0.47       708
   macro avg       0.48      0.46      0.41       708
weighted avg       0.71      0.47      0.54       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.4911
         Training Macro Recall = 0.4912
    Training Balanced Accuracy = 0.4912

****************************** Test Scores ******************************
                Test Macro F1 = 0.4065
             Test Macro Recall = 0.4629
        Test Balanced Accuracy = 0.4629

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.08460000000000001
       Train-Test Macro Recall Diff = -0.028300000000000047
Train-Test Balanced Accuracy Diff = -0.028300000000000047

****************************** Graphs for Test ******************************
```



ROS - LR no stemming/lemmas

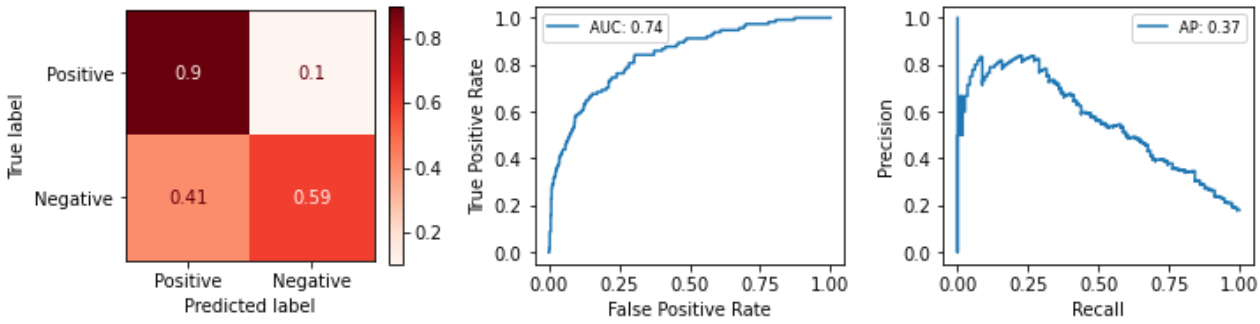
I'm going to focus on Logistic Regression models, since they performed the best in terms of cross-validated recall scores. I don't have the resources to redo a full gridsearch, so I'll use the same best parameters we got from gridsearching unsampled data.

In [413…

```
# copy the best LR pipeline/model from the gridsearch (deep copy)
ros_LR_binary_nostemlem_pipe = clone(best_LR_binary_nostemlem_pipe)

# get cross_val_scores for best LR model with stemming/lemmatization
xval = cross_val_score(ros_LR_binary_nostemlem_pipe, X2ros_train, y2ros_train,
                       scoring='recall_macro', n_jobs=-1, verbose=True)

print(f"All scores: {xval}")
print(f"Average x-validated score (recall): {np.round(np.mean(xval),3)}")
print()

# fit model to generate report
ros_LR_binary_nostemlem_pipe.fit(X2ros_train, y2ros_train)
```

```
eval_clf_model(ros_LR_binary_nostemlem_pipe, X2_test, y2_test, X2ros_train,
               y2ros_train, labels=class_labels, normalize_cm='true')
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    2.9s finished
All scores: [0.91052632 0.91157895 0.92421053 0.92734399 0.93568066]
Average x-validated score (recall): 0.922

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.99      0.95      0.97      2374
           1       0.95      0.99      0.97      2374

    accuracy                           0.97      4748
   macro avg       0.97      0.97      0.97      4748
weighted avg       0.97      0.97      0.97      4748


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.92      0.90      0.91       594
           1       0.52      0.59      0.55       114

    accuracy                           0.85       708
   macro avg       0.72      0.74      0.73       708
weighted avg       0.86      0.85      0.85       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.9699
         Training Macro Recall = 0.9699
    Training Balanced Accuracy = 0.9699

****************************** Test Scores ******************************
                Test Macro F1 = 0.7309
             Test Macro Recall = 0.7425
        Test Balanced Accuracy = 0.7425

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.239
       Train-Test Macro Recall Diff = -0.22739999999999994
Train-Test Balanced Accuracy Diff = -0.22739999999999994

****************************** Graphs for Test ******************************
```



Still tending to overfit, but better performance in terms of Test Balanced Accuracy and Recall. I think cross-validated recall macro is naturally higher here because I've balanced the classes.

I think these will end up being the best models. I'll go ahead and run the models with previously determined best params on the oversampled training data and see what we get.

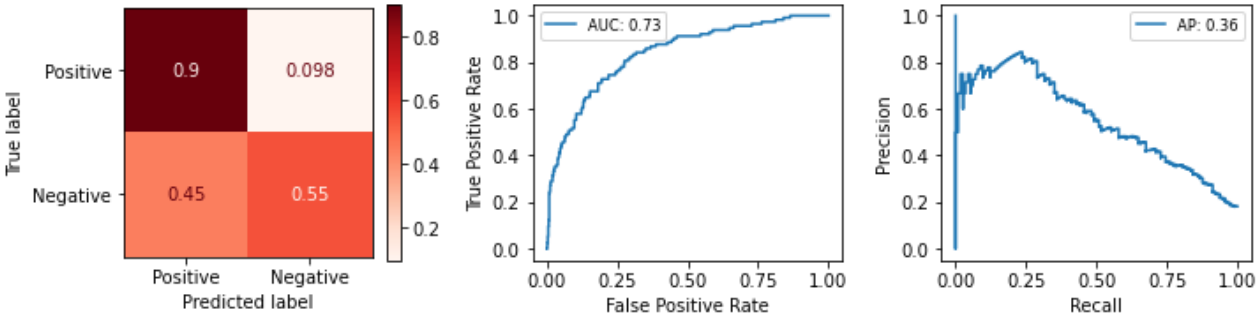
Try with `min_df=2` ? I read that removing words with low frequencies could help with overfitting.

In [414…

```
# copy the best LR pipeline/model from the gridsearch (deep copy)
ros_LR_binary_min_nostemlem_pipe = clone(best_LR_binary_nostemlem_pipe)

# try setting min_df=2 (default is 1) to remove tokens with freq 2 or less
# perhaps it will overfit less?
ros_LR_binary_min_nostemlem_pipe.set_params(prep__vect__min_df=2)

# get cross_val_scores for best MNB model with stemming/lemmatization
xval = cross_val_score(ros_LR_binary_min_nostemlem_pipe, X2ros_train, y2ros_trai
                       scoring='recall_macro', n_jobs=-1, verbose=True)

print(f"All scores: {xval}")
print(f"Average x-validated score (recall): {np.round(np.mean(xval),3)}")
print()

# fit model to generate report
ros_LR_binary_min_nostemlem_pipe.fit(X2ros_train, y2ros_train)

eval_clf_model(ros_LR_binary_min_nostemlem_pipe, X2_test, y2_test, X2ros_train,
               y2ros_train, labels=class_labels, normalize_cm='true')
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    2.4s finished
All scores: [0.90947368 0.91473684 0.92526316 0.9315523  0.93884966]
Average x-validated score (recall): 0.924

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.99      0.95      0.97      2374
           1       0.95      0.99      0.97      2374

    accuracy                           0.97      4748
   macro avg       0.97      0.97      0.97      4748
weighted avg       0.97      0.97      0.97      4748


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       594
           1       0.52      0.55      0.54       114

    accuracy                           0.85       708
   macro avg       0.72      0.73      0.72       708
weighted avg       0.85      0.85      0.85       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.9684
         Training Macro Recall = 0.9684
    Training Balanced Accuracy = 0.9684

****************************** Test Scores ******************************
                Test Macro F1 = 0.7219
             Test Macro Recall = 0.7275
        Test Balanced Accuracy = 0.7275

****************************** Differences ******************************
```

```
        Train-Test Macro F1 Diff = -0.24650000000000005
     Train-Test Macro Recall Diff = -0.2409
Train-Test Balanced Accuracy Diff = -0.2409

****************************** Graphs for Test ******************************
```



No, I tried a few different values (0, 2, 5) and none made much difference, except that 5 made performance on Negative class worse.

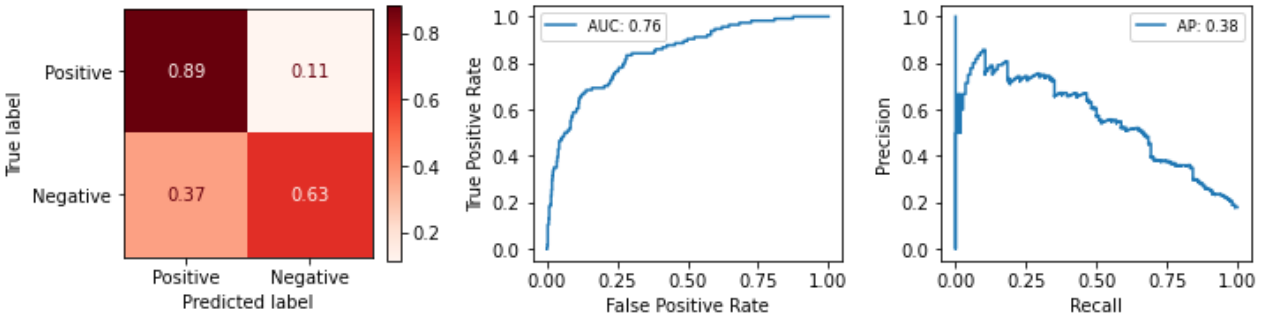
ROS - LR with stemming/lemmas

In [415…

```
# copy the best MNB pipeline/model from the gridsearch (deep copy)
ros_LR_binary_withstemlem_pipe = clone(best_LR_binary_nostemlem_pipe)

# set tokenizer in the copy to use the function with stemming and lemmatization
ros_LR_binary_withstemlem_pipe.set_params(prep__vect__tokenizer=tokenize_lemma_s

# get cross_val_scores for best MNB model with stemming/lemmatization
xval = cross_val_score(ros_LR_binary_withstemlem_pipe, X2ros_train, y2ros_train,
                       scoring='recall_macro', n_jobs=-1, verbose=True)

print(f"All scores: {xval}")
print(f"Average x-validated score (recall): {np.round(np.mean(xval),3)}")
print()

# fit model to generate report
ros_LR_binary_withstemlem_pipe.fit(X2ros_train, y2ros_train)

eval_clf_model(ros_LR_binary_withstemlem_pipe, X2_test, y2_test, X2ros_train,
               y2ros_train, labels=class_labels)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:   11.9s finished
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/feature_extraction/text.py:383: UserWarning: Your stop_words may be incons
istent with your preprocessing. Tokenizing the stop words generated tokens ["i
t'"] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
All scores: [0.91684211 0.90105263 0.91789474 0.92733511 0.93884744]
Average x-validated score (recall): 0.92

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.99      0.94      0.96      2374
           1       0.94      0.99      0.97      2374

    accuracy                           0.96      4748
   macro avg       0.97      0.96      0.96      4748
weighted avg       0.97      0.96      0.96      4748
```

```
****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.93      0.89      0.91       594
           1       0.51      0.63      0.57       114

    accuracy                           0.84       708
   macro avg       0.72      0.76      0.74       708
weighted avg       0.86      0.84      0.85       708


****************************** Training Scores ******************************
            Training Macro F1 = 0.9644
         Training Macro Recall = 0.9644
    Training Balanced Accuracy = 0.9644

****************************** Test Scores ******************************
                Test Macro F1 = 0.7361
             Test Macro Recall = 0.7586
        Test Balanced Accuracy = 0.7586

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.22830000000000006
       Train-Test Macro Recall Diff = -0.20579999999999998
Train-Test Balanced Accuracy Diff = -0.20579999999999998

****************************** Graphs for Test ******************************
```



Preprocessing for multi ROS

In [416…
```
# train test split
X2m_train, X2m_test, y2m_train, y2m_test = train_test_split(Xm, ym, test_size=0.
                                                    stratify=ym)
print(len(X2m_train))
print(len(y2m_train))
print(len(X2m_test))
print(len(y2m_test))
```

```
7127
7127
1782
1782
```

In [417…
```
X2m_train = pd.DataFrame(X2m_train, columns=['cleaned'])
```

In [418…
```
X2mros_train, y2mros_train = ros.fit_resample(X2m_train,
                                              y2m_train)
```

In [419…
```
y2mros_train.value_counts()
```

Out[419…
```
2    4298
1    4298
0    4298
Name: emotion, dtype: int64
```

In [420…
```
print(len(X2mros_train))
print(len(y2mros_train))
```

```
12894
12894
```

In [421…
```
print(X2mros_train.shape)
print(y2mros_train.shape)
```

```
(12894, 1)
(12894,)
```

In [422…
```
X2mros_train = pd.Series(data=X2mros_train['cleaned'])
X2mros_train.shape
```

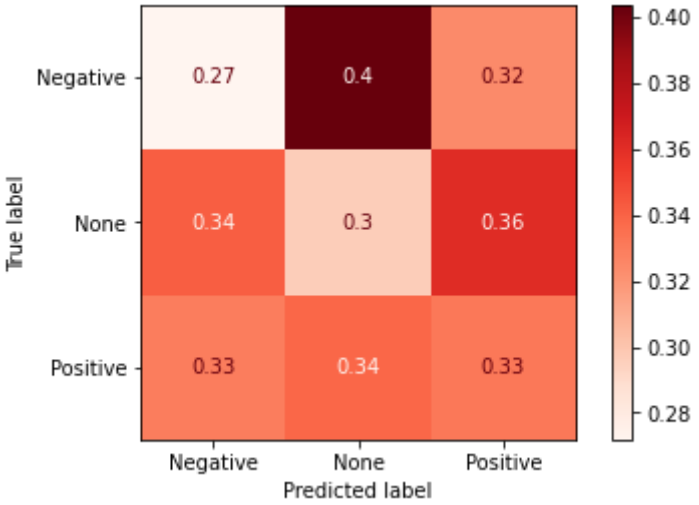
Out[422…
```
(12894,)
```

ROS - Dummy Classifier

In [423…
```
# pre-processing pipeline to transform into vectors
prep_pipe = Pipeline([
    ('vect', CountVectorizer(tokenizer=tweettokenizer.tokenize)),
    ('trans', TfidfTransformer())
])

# Let's make a baseline classifier using the dummy model
clf_pipe = Pipeline([
    ('prep', prep_pipe),
    ('clf', DummyClassifier(strategy='stratified'))
])

clf_pipe.fit(X2mros_train, y2mros_train)

eval_clf_model(clf_pipe, X2m_test, y2m_test, X2mros_train, y2mros_train,
               labels=mclass_labels)
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.33      0.33      0.33      4298
           1       0.33      0.33      0.33      4298
           2       0.33      0.33      0.33      4298

    accuracy                           0.33     12894
   macro avg       0.33      0.33      0.33     12894
weighted avg       0.33      0.33      0.33     12894


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.05      0.27      0.09       114
           1       0.58      0.30      0.40      1074
           2       0.34      0.35      0.35       594

    accuracy                           0.32      1782
```

```
   macro avg       0.32      0.31      0.28      1782
weighted avg       0.47      0.32      0.36      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.3281
         Training Macro Recall = 0.3281
    Training Balanced Accuracy = 0.3281

****************************** Test Scores **********************************
                Test Macro F1 = 0.2776
             Test Macro Recall = 0.3094
         Test Balanced Accuracy = 0.3094

****************************** Differences **********************************
          Train-Test Macro F1 Diff = -0.05049999999999999
       Train-Test Macro Recall Diff = -0.018699999999999994
Train-Test Balanced Accuracy Diff = -0.018699999999999994

****************************** Graphs for Test ******************************
```



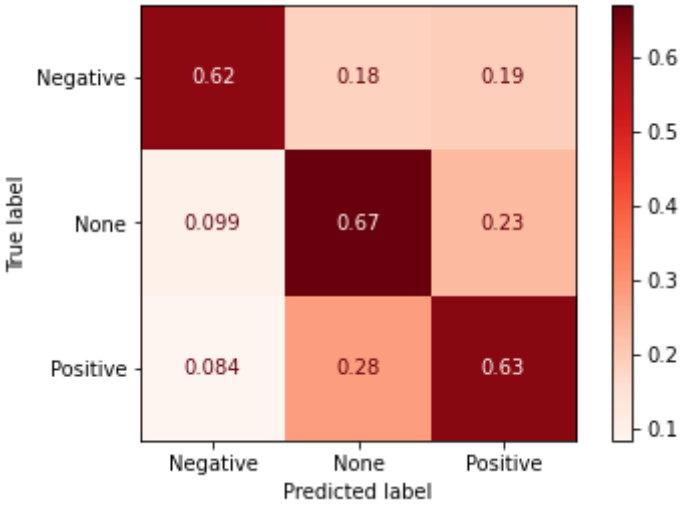
ROS - LR no stemming/lemmas

In [424…

```
# copy the best MNB pipeline/model from the gridsearch (deep copy)
ros_LR_multiclass_nostemlem_pipe = clone(best_LR_multiclass_nostemlem_pipe)

# get cross_val_scores for best MNB model with stemming/lemmatization
xval = cross_val_score(ros_LR_multiclass_nostemlem_pipe, X2mros_train, y2mros_tr
                       scoring='recall_macro', n_jobs=-1, verbose=True)

print(f"All scores: {xval}")
print(f"Average x-validated score (recall): {np.round(np.mean(xval),3)}")
print()

# fit model to generate report
ros_LR_multiclass_nostemlem_pipe.fit(X2mros_train, y2mros_train)

eval_clf_model(ros_LR_multiclass_nostemlem_pipe, X2m_test, y2m_test, X2mros_trai
               y2mros_train, labels=mclass_labels)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:   10.5s finished
All scores: [0.75191272 0.74403353 0.76884921 0.79487686 0.77934907]
Average x-validated score (recall): 0.768
```

```
****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      4298
           1       0.81      0.80      0.81      4298
           2       0.85      0.81      0.83      4298

    accuracy                           0.86     12894
   macro avg       0.86      0.86      0.86     12894
weighted avg       0.86      0.86      0.86     12894


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.31      0.62      0.42       114
           1       0.79      0.67      0.73      1074
           2       0.58      0.63      0.61       594

    accuracy                           0.65      1782
   macro avg       0.56      0.64      0.58      1782
weighted avg       0.69      0.65      0.67      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.8588
         Training Macro Recall = 0.8599
    Training Balanced Accuracy = 0.8599

****************************** Test Scores ******************************
                Test Macro F1 = 0.5825
             Test Macro Recall = 0.6415
        Test Balanced Accuracy = 0.6415

****************************** Differences ******************************
          Train-Test Macro F1 Diff = -0.2763
       Train-Test Macro Recall Diff = -0.21840000000000004
Train-Test Balanced Accuracy Diff = -0.21840000000000004

****************************** Graphs for Test ******************************
```



Test balanced accuracy is a bit higher, although not much considering how much variation I've been seeing based on the train-test-splits.

Fewer Nones and Positives seem to be misclassified as Negative in this model, compared to the unsampled one.

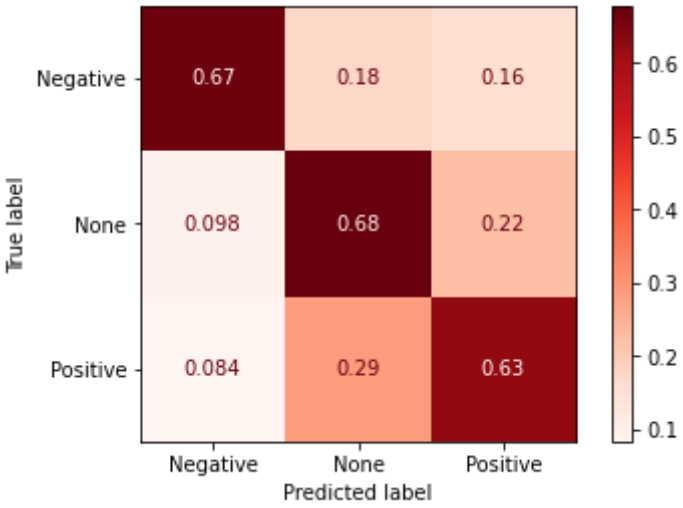
ROS - LR with stemming/lemmas

In [425…

```
# copy the best MNB pipeline/model from the gridsearch (deep copy)
ros_LR_multiclass_withstemlem_pipe = clone(best_LR_multiclass_nostemlem_pipe)

# set tokenizer in the copy to use the function with stemming and lemmatization
ros_LR_multiclass_withstemlem_pipe.set_params(prep__vect__tokenizer=tokenize_lem

# get cross_val_scores for best MNB model with stemming/lemmatization
xval = cross_val_score(ros_LR_multiclass_withstemlem_pipe, X2mros_train, y2mros_
                       scoring='recall_macro', n_jobs=-1, verbose=True)

print(f"All scores: {xval}")
print(f"Average x-validated score (recall macro): {np.round(np.mean(xval),3)}")
print()

# fit model to generate report
ros_LR_multiclass_withstemlem_pipe.fit(X2mros_train, y2mros_train)

eval_clf_model(ros_LR_multiclass_withstemlem_pipe, X2m_test, y2m_test, X2mros_tr
               y2mros_train, labels=mclass_labels)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:   29.4s finished
/Users/jessicamiles/opt/anaconda3/envs/learn-env2/lib/python3.8/site-packages/sk
learn/feature_extraction/text.py:383: UserWarning: Your stop_words may be incons
istent with your preprocessing. Tokenizing the stop words generated tokens ["i
t'"] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
All scores: [0.75230076 0.74675168 0.76535858 0.7890602  0.78128615]
Average x-validated score (recall macro): 0.767

****************************** Training Data ******************************
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      4298
           1       0.81      0.79      0.80      4298
           2       0.83      0.80      0.82      4298

    accuracy                           0.85     12894
   macro avg       0.85      0.85      0.85     12894
weighted avg       0.85      0.85      0.85     12894


****************************** Test Data ******************************
              precision    recall  f1-score   support

           0       0.33      0.67      0.44       114
           1       0.79      0.68      0.73      1074
           2       0.59      0.63      0.61       594

    accuracy                           0.66      1782
   macro avg       0.57      0.66      0.59      1782
weighted avg       0.69      0.66      0.67      1782


****************************** Training Scores ******************************
            Training Macro F1 = 0.853
         Training Macro Recall = 0.8541
   Training Balanced Accuracy = 0.8541

****************************** Test Scores ******************************
                Test Macro F1 = 0.5927
```

```
            Test Macro Recall = 0.6569
        Test Balanced Accuracy = 0.6569

****************************** Differences ******************************
         Train-Test Macro F1 Diff = -0.2603
     Train-Test Macro Recall Diff = -0.19719999999999993
Train-Test Balanced Accuracy Diff = -0.19719999999999993

****************************** Graphs for Test ******************************
```



Best Models

The best binary and multi-class models I'm selecting are the ones from this run where I used Random Oversampling, and the Logistic Regression models without applying stemming or lemmatization.

In [140…
```
# pull out the best binary model from the Logistic Regression pipeline
best_binary = ros_LR_binary_nostemlem_pipe.named_steps['clf']

# save out best binary model
model_path = 'models/'
joblib.dump(best_binary, f"{model_path}best_binary_model.joblib.gz")

# pull out the best multi-class model from the Logistic Regression pipeline
best_multi = ros_LR_multiclass_nostemlem_pipe.named_steps['clf']

# save out best model for later, if needed
joblib.dump(best_multi, f"{model_path}best_multiclass_model.joblib.gz")
```

Out[140…
```
['models/best_multiclass_model.joblib.gz']
```

In terms of best parameters, the best models agreed on most things but differed in a few ways:

Type	Doc Term Matrix Freq	Ngrams Range	Stopwords removal	Model params
Binary	TF-IDF	(1, 1)	Custom stopwords (minimal) and all punctuation except ? and !	C=1, fit_intercept=False, solver=newton-cg
Multi-class	TF-IDF	(1, 2)	Custom stopwords (minimal) and all punctuation except ? and !	C=0.1, fit_intercept=True, solver=newton-cg

iNTERPRET

Evaluate how well your work solves the stated business problem.

Questions to consider:

- How do you interpret the results?
- How well does your model fit your data? How much better is this than your baseline model?
- How confident are you that your results would generalize beyond the data you have?
- How confident are you that this model would benefit the business if put into use?

Binary Coefficients

Let's review the coefficients for the binary classification problem first.

In [142…
```
# get the feature names representing the words from the best pipeline
feature_names = ros_LR_binary_nostemlem_pipe.named_steps['prep'].\
    named_steps['vect'].get_feature_names()
feature_names[:5]
```

Out[142…
```
['!', '##sxsw', '#10', '#100tc', '#106']
```

In [143…
```
# convert logit coefficients from log odds to odds ratios
lr_odds = np.exp(best_binary.coef_[0])
lr_odds_s = pd.Series(lr_odds, index=feature_names)
lr_odds_s.sort_values(ascending=False)[:10]
```

Out[143…
```
not          24.453692
headaches    21.196614
long         13.755613
#fail        11.634709
hate         10.044353
why           9.513645
money         9.148775
no            8.989046
sucks         8.547767
doesn't       8.032493
dtype: float64
```

Negative

In [144…
```
# Create dataframe for odds greater than 1 (Negative).
# Higher values represent greater odds of being negative
bin_neg = np.round(lr_odds_s.loc[lr_odds_s >= 1], 2).to_frame(
    name='Odds of Negative')
bin_neg.reset_index(inplace=True)
bin_neg.rename(columns={'index': 'Word'}, inplace=True)

# sort the dataframe by frequency
bin_neg.sort_values(by='Odds of Negative', ascending=False, inplace=True)
bin_neg.head()
```

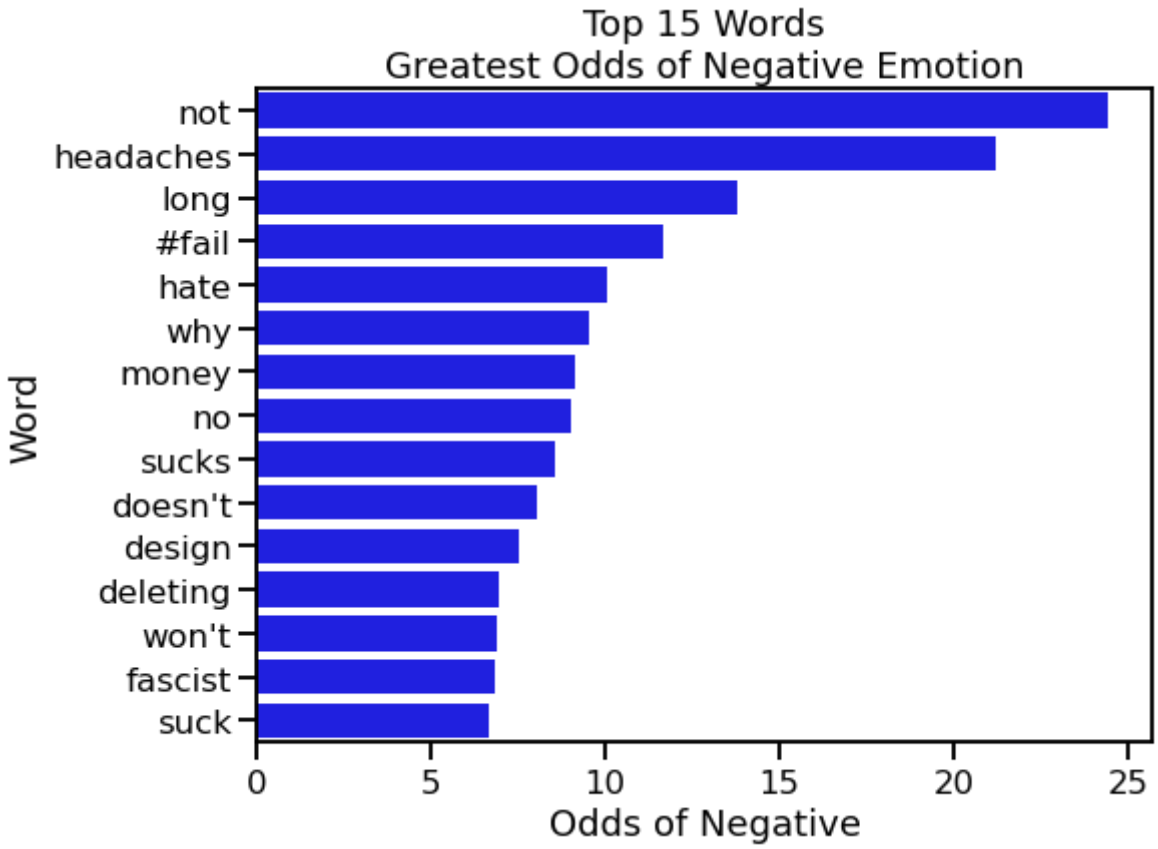
Out[144…

	Word	Odds of Negative

	Word	Odds of Negative
982	not	24.45
713	headaches	21.20
872	long	13.76
30	#fail	11.63
711	hate	10.04

In [145…
```
# Replace naughty words since I will be sharing this!
bin_neg['Word'].replace('fuck', 'f**k', inplace=True)
bin_neg['Word'].replace('fucking', 'f**king', inplace=True)
```

In [146…
```
# Create visualizations for words with highest odds of negative emotion
generate_freqs_wordcloud(bin_neg, 'Word', 'Odds of Negative', cmap="OrRd",
                         min_font_size=16)

plot_wordfreqs(bin_neg, 'Word', 'Odds of Negative', 15,
                      "Greatest Odds of Negative Emotion")
```



These definitely look like words associated with negative emotions.

- `#fail` is a pretty general hashtag, but if I were working for Apple or Google, I would want to review tweets with my brand or product name in it as well as this hashtag to learn more
- `headaches` and `design` stand out as potentially being related to some feature or function of a device not working as expected
- `deleting` sounds like it might be related to an app, so I would check into what product or app is mentioned in these tweets
- `iphone` is in this word cloud, so Apple would probably want to review the tweets classified as negative and containing `iphone` to see what else is mentioned
- `long` features in both the top 15 words and word cloud. From what I've read while processing these, I'd guess this is referring to long lines that people described to get into an Apple pop-up store in Austin.

Positive

In [147…

```
# Create dataframe for odds less than 1.
# Higher values represent greater odds of being positive
bin_pos = np.round(1/(lr_odds_s.loc[lr_odds_s < 1]), 2).to_frame(
    name='Odds of Positive')
bin_pos.reset_index(inplace=True)
bin_pos.rename(columns={'index': 'Word'}, inplace=True)

# sort the dataframe by frequency
bin_pos.sort_values(by='Odds of Positive', ascending=False, inplace=True)
bin_pos.head()
```

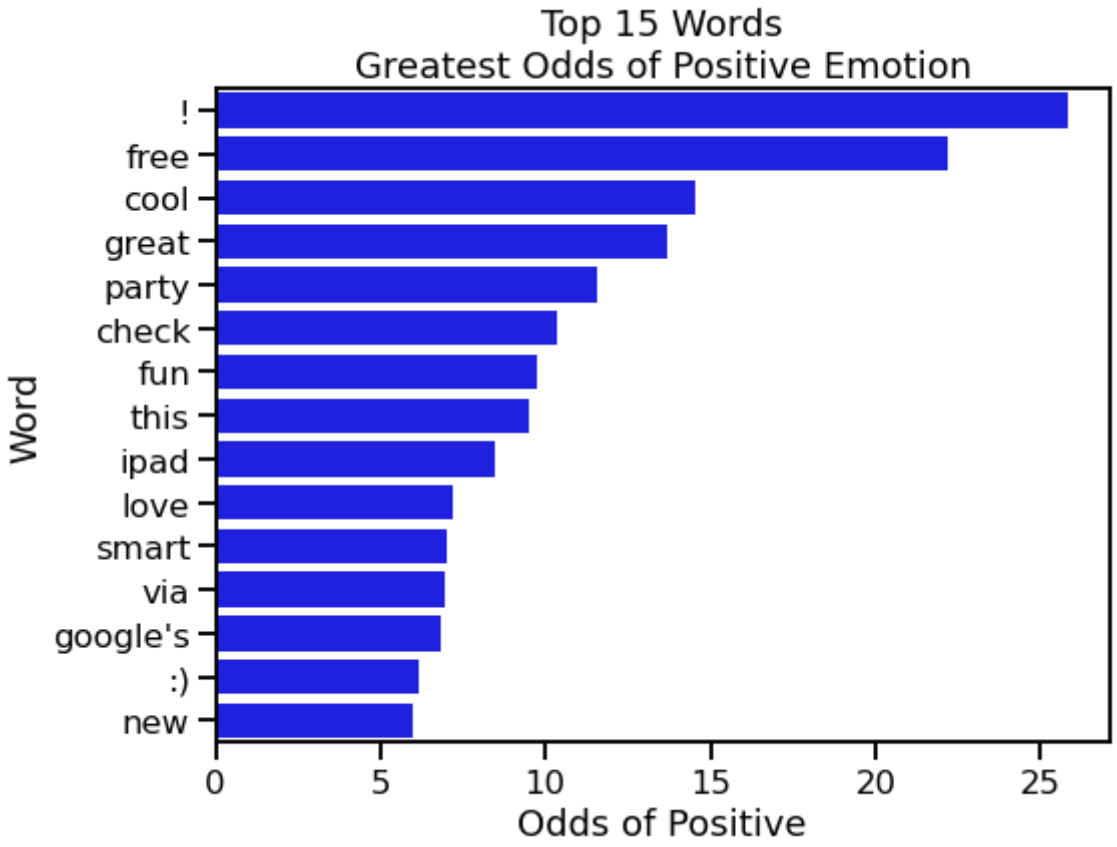
Out[147…

	Word	Odds of Positive

	Word	Odds of Positive
0	!	25.82
1773	free	22.19
1246	cool	14.57
1911	great	13.68
2823	party	11.58

In [148…

```
# Create visualizations for words with highest odds of positive emotion
generate_freqs_wordcloud(bin_pos, 'Word', 'Odds of Positive', cmap="Greens",
                         min_font_size=16)

plot_wordfreqs(bin_pos, 'Word', 'Odds of Positive', 15,
                       "Greatest Odds of Positive Emotion")
```



And these definitely look like words associated with positive emotions.

- Perhaps unsurprisingly, tweets with an exclamation point greatly increase the odds of positive emotions.
- `free` is the next most influential word for positive emotion. This is probably not surprising either, but if companies are looking for what gets people to express positive emotion on Twitter, free stuff seems like a good idea.
- `ipad` makes it into both the word cloud and the top 15 words, so in this corpus people are being positive about the ipad, whereas they were tending to be more negative towards the iphone.
- `store` and `popup` are both in the word cloud. While there were probably some people tweeting negatively about the long line, there were also others who tweeted positively about the popup store.

Multi-class Coefficients

In addition to reviewing the words that indicate highest odds of 'No emotion towards brand or product', I'm also interested in whether the models will agree on what tokens contribute the highest odds to Positive and Negative.

In [149…
```
# get the feature names representing the words from the best pipeline
feature_names = ros_LR_multiclass_nostemlem_pipe.named_steps['prep'].\
    named_steps['vect'].get_feature_names()
feature_names[:5]
```

Out[149… `['!', '! !', '! #agchat', '! #android', '! #androidsxsw']`

Negative

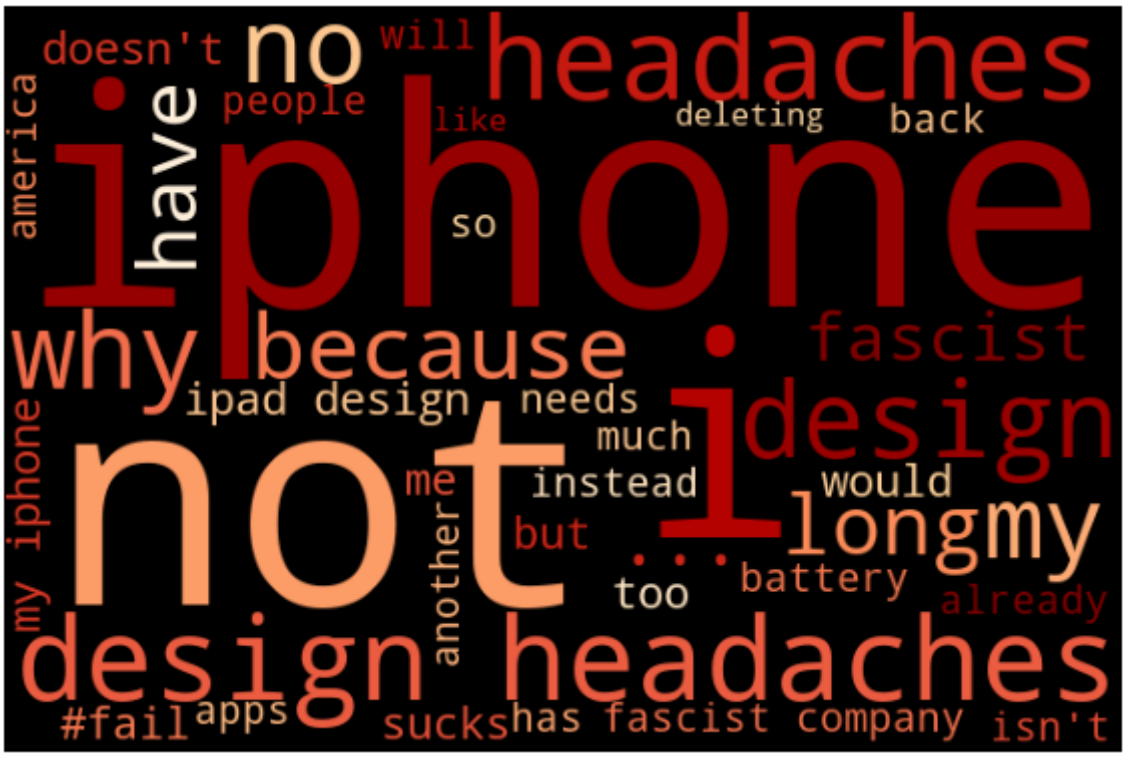
In [150…
```
# convert logit coefficients from log odds to odds ratios
# index 0 is negative
lr_odds = np.exp(best_multi.coef_[0])
lr_odds_s = pd.Series(lr_odds, index=feature_names)

# Create dataframe for negative odds
multi_neg = np.round(lr_odds_s, 2).to_frame(name='Odds of Negative')
multi_neg.reset_index(inplace=True)
multi_neg.rename(columns={'index': 'Word'}, inplace=True)

# sort the dataframe by frequency
multi_neg.sort_values(by='Odds of Negative', ascending=False, inplace=True)
multi_neg.head()
```

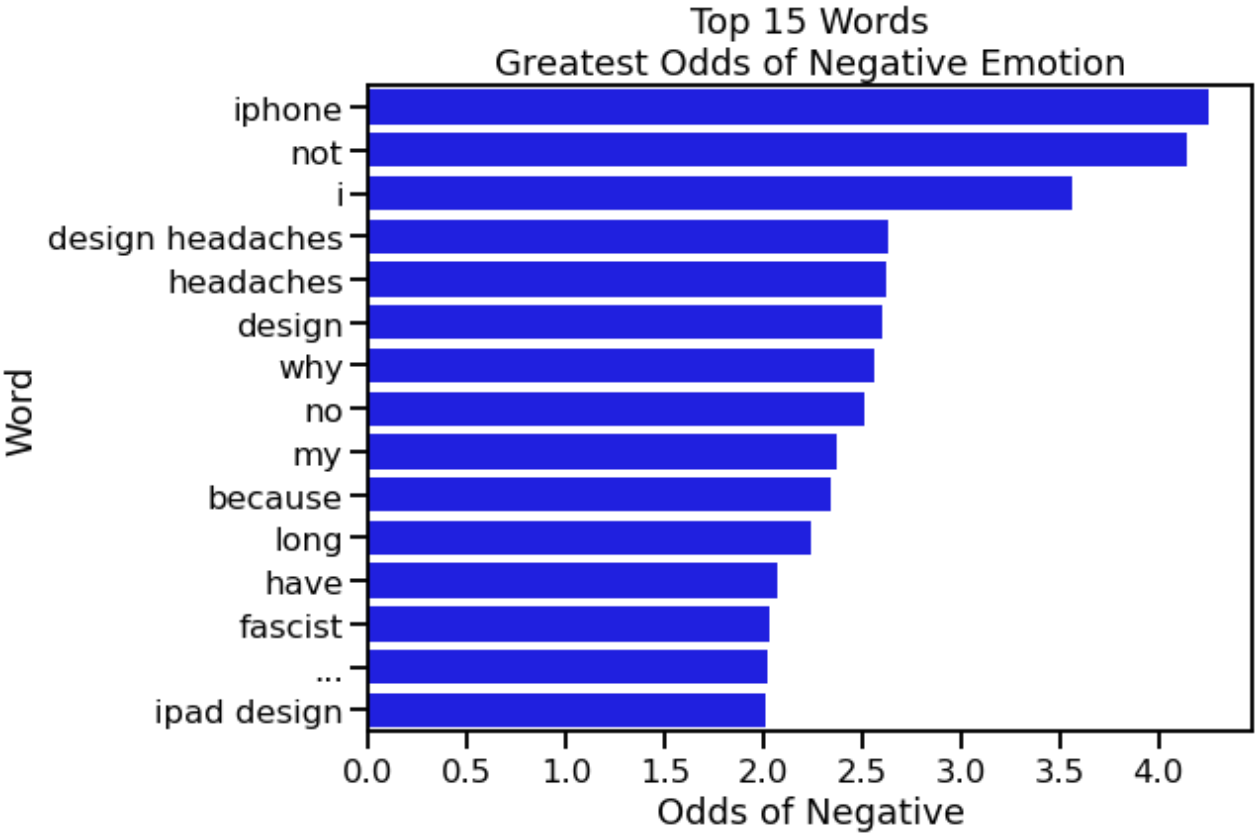
Out[150…

	Word	Odds of Negative
24062	iphone	4.26
30584	not	4.15
22175	i	3.56
13765	design headaches	2.63
20980	headaches	2.62

In [151…
```
# Replace naughty words since I will be sharing this!
multi_neg['Word'].replace('fuck', 'f**k', inplace=True)
multi_neg['Word'].replace('fucking', 'f**king', inplace=True)
```

In [152…
```
# Create visualizations for words with highest odds of negative emotion
generate_freqs_wordcloud(multi_neg, 'Word', 'Odds of Negative', cmap="OrRd",
                         min_font_size=14)

plot_wordfreqs(multi_neg, 'Word', 'Odds of Negative', 15,
               "Greatest Odds of Negative Emotion")
```





These mostly look like negative words, although `my` and `i` are pretty neutral. Most of these are familiar from the binary model, but the weights are different.

- `iphone` is the word with the greatest odds of negative emotion, which is interesting. Apple would definitely want to look into that. It didn't even make it into the top 15 in the binary mode..

- `#fail` was the the second word in the top 15 for the binary model, but it's much less influential in the multi-class model.
- `headaches` and `design` were in the binary model, and here where we had bigrams as well as unigrams we can see `design headaches` as well as `ipad design`. This is more informative than the unigrams.

Positive

In [153…
```
# convert logit coefficients from log odds to odds ratios
# index 2 is positive
lr_odds = np.exp(best_multi.coef_[2])
lr_odds_s = pd.Series(lr_odds, index=feature_names)

# Create dataframe for positive odds
multi_pos = np.round(lr_odds_s, 2).to_frame(name='Odds of Positive')
multi_pos.reset_index(inplace=True)
multi_pos.rename(columns={'index': 'Word'}, inplace=True)

# sort the dataframe by frequency
multi_pos.sort_values(by='Odds of Positive', ascending=False, inplace=True)
multi_pos.head()
```

Out[153…

	Word	Odds of Positive
0	!	9.20
12462	cool	2.49
19826	great	2.37
23464	ipad	2.18
8078	awesome	2.08

In [154…
```
# Create visualizations for words with highest odds of positive emotion
generate_freqs_wordcloud(multi_pos, 'Word', 'Odds of Positive', cmap="Greens",
                         min_font_size=16)

plot_wordfreqs(multi_pos, 'Word', 'Odds of Positive', 15,
               "Greatest Odds of Positive Emotion")
```

These seem to agree with the binary model pretty well. Although, only the exclamation point has very high odds; the odds of positive emotion for the next word `ipad` is only 2.5. In the binary mode, odds of positive for `ipad` are about 10.

- An exclamation point is still the top predictor of positive emotion.
- `free` doesn't make it into the top 15 here, although I see it pretty small in the word cloud.
- `ipad` is the second highest predictor of positive

- I see more brand names in this positive word cloud, such as `apple` , and `google party` .

No emotion

In [155…

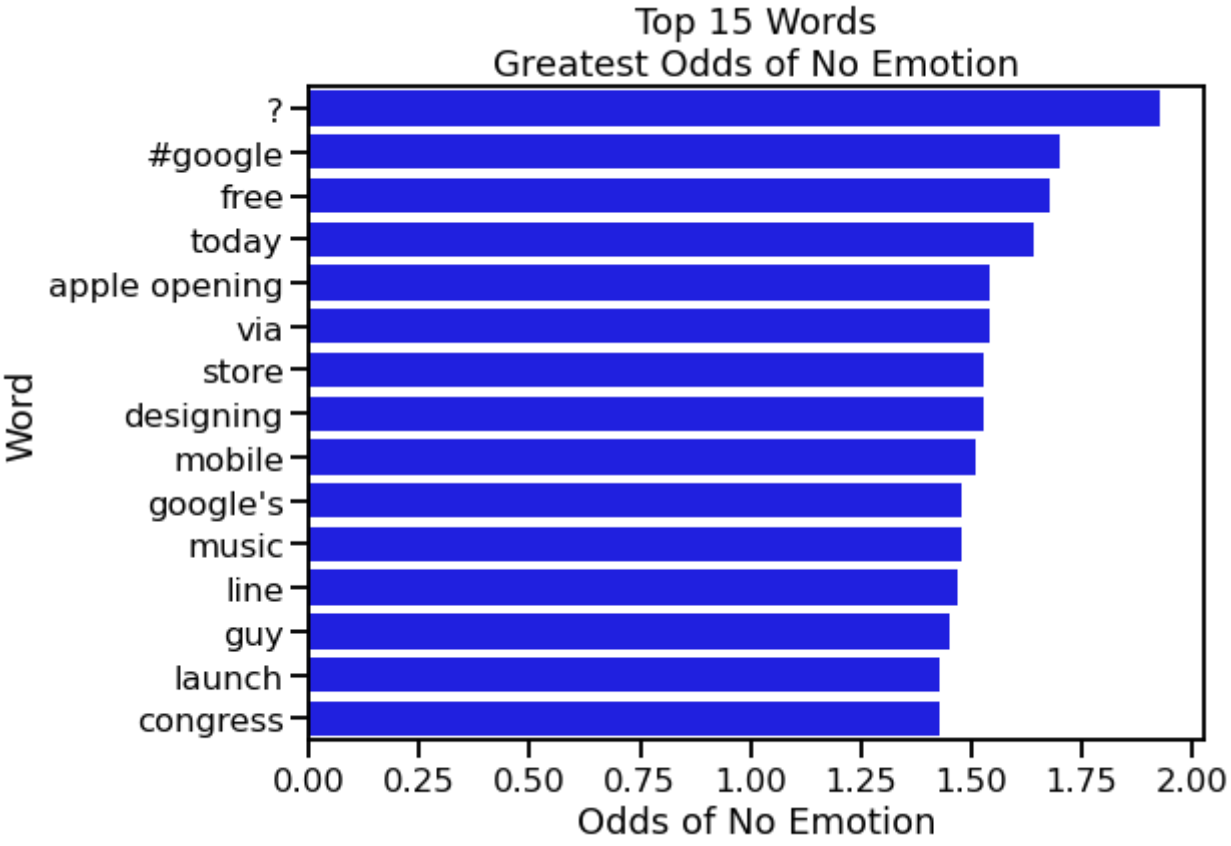
```
# convert logit coefficients from log odds to odds ratios
# index 1 is no emotion
lr_odds = np.exp(best_multi.coef_[1])
lr_odds_s = pd.Series(lr_odds, index=feature_names)

# Create dataframe for no emotion odds
# only odds over 1, since anything less than 1 is not more likely
multi_none = np.round(lr_odds_s.loc[lr_odds_s >= 1], 2).\
    to_frame(name='Odds of No Emotion')
multi_none.reset_index(inplace=True)
multi_none.rename(columns={'index': 'Word'}, inplace=True)

# sort the dataframe by frequency
multi_none.sort_values(by='Odds of No Emotion', ascending=False, inplace=True)
multi_none.head()
```

Out[155…

	Word	Odds of No Emotion
2773	?	1.93
883	#google	1.70
10037	free	1.68
23752	today	1.64
4222	apple opening	1.54

In [156…

```
# Create visualizations for words with highest odds of positive emotion
generate_freqs_wordcloud(multi_none, 'Word', 'Odds of No Emotion', cmap="Blues",
                         min_font_size=16)

plot_wordfreqs(multi_none, 'Word', 'Odds of No Emotion', 15,
               "Greatest Odds of No Emotion")
```



- Some of the words that were associated with positive emotion in the binary model are now associated with No emotion in the multi-class model. Examples include `free` and `store`.
- `line` was associated with negative emotion in the binary model, but here is no emotion.
- We definitely have brands being mentioned such as `google`, `apple`. That must mean that people are mentioning brands but not expressing any emotion that the people who labeled the dataset could discern.

Investigating the labeling

An important factor in interpreting these results the quality of the labeling used to train the models.

This corpus was labeled by CrowdFlower (now called Figure Eight), meaning human beings did the labeling.

- I was not able to determine who made up the group of labels (whether they do this professionally, were volunteers, their level of English reading comprehension, etc.)
- I also could not determine what, if any, steps were taken after the labeling to perform quality control checks or standardization on the accuracy and consistency of the labels applied. Since there were some still left labeled 'I can't tell' I think it's likely there was no QC step.
- We don't know how clear the instructions to the labelers were. Deciding sentiment alone is fairly easy, but the labels here are somewhat convoluted since people need to put positive or negative sentiment that is NOT towards a brand or product, AND neutral sentiment towards a brand or product in a single category. Without clear instructions, some people may have simply tagged based on sentiment.
- Finally, I have definitely noticed some tweets labeled as having sentiment towards a brand or product, but the sentiment is directed towards a third party iPhone or Android app, not a core function of the operating system or even an app developed by Apple or Google. I would imagine that representatives from Apple or Google probably value feedback about the products and services they actually put out, versus apps that run on their products, and over which they have little to no quality control.

Although human beings are certainly better at detecting sentiment and emotion than computer systems, there are no objective standards for deciding sentiment and emotion. In other words, the ground truth represented by these labels is still fairly subjective, adding an additional layer of noise to the data on top of the existing noise related to general differences in how people use words.

It is not only the interpretation of sentiment which is subjective and thus prone to noisy labeling by the humans who classify the data. Sentiment is also inherently noisy in general because it is not a black or white concept; highly positive or highly negative. Both positive and negative sentiments may be expressed on gradient scales, with a very muddy neutral center.

In [157…

```
# let's get a sample of "Positive"
df.loc[df['emotion']=="Positive emotion",['tweet_text', 'product', 'emotion']][4
```

Out[157…

	tweet_text	product	emotion
95	Yai!!! RT @mention New #UberSocial for #iPhone now in the App Store includes UberGuide to #SXSW sponsored by (cont) {link}	iPhone	Positive emotion
97	Fast, Fun & Future: @mention of Google presenting at #sxsw on search, local and mobile	Google	Positive emotion
100	Headline: "#iPad 2 is the Must-Have Gadget at #SXSW" Hmm... I could have seen that one coming! {link} #gadget	iPad	Positive emotion

	tweet_text	product	emotion
103	.@mention "Google launched checkins a month ago." Check ins are ok, but CHECK OUTS are the future. #sxsw #Bizzy	Google	Positive emotion
105	‰ÛÏ@mention "Google before you tweet" is the new "think before you speak." - Mark Belinsky, #911tweets panel at #SXSW.‰Û	Google	Positive emotion
108	Kawasaki: "Not C.S. Lewis level reasoning, but Apple's continued existence is evidence for the existence of God" #bawling #sxsw	Apple	Positive emotion
110	Kawasaki: "pagemaker saved Apple." Oh those were the days. #sxsw #jwtatl #enchantment via @mention	Apple	Positive emotion
111	Spark for #android is up for a #teamandroid award at #SXSW read about it here: {link}	NaN	Positive emotion
113	#SXSW and #Apple iPad 2's are great, but thoughts are w/ Japan and APAC regions dealing w/ earthquake & tsunami trauma. #sxswi	iPad	Positive emotion
115	At #SXSW, #Apple schools the marketing experts - {link}	Apple	Positive emotion

I agree that some of these are positive, but definitely not all.

- I agree with the positive labels for rows 95, 97, 100, 111, 113, and 115. The rest, I think are pretty neutral towards the actual brand or product.
- Rows 103 and 105 may even be somewhat negative towards Google.

So I only agree with 6 out of these 10 in the selected sample.

In [158…
```
# let's get a sample of "Negative"
df.loc[df['emotion']=="Negative emotion",['tweet_text', 'product', 'emotion']][4
```

Out[158…

	tweet_text	product	emotion
695	shoot, my ipad will not display any search results :(will have to go through questions later #osmpw #sxsw	iPad	Negative emotion
698	The #SXSW iPhone app is mocking me. Like I have the money to be there - COME ON.	iPad or iPhone App	Negative emotion
699	The #SXSW iPhone app is one of the worst I've had to use in a very long time.	iPad or iPhone App	Negative emotion
706	Disliking iPhone twitter auto shortening links for me. #sxsw	iPad or iPhone App	Negative emotion
712	my iPhone is overheating. why are there so many british sounding people in texas? #SXSW	iPhone	Negative emotion
720	My iPhone is wilting under the stress of being at #sxsw.	iPhone	Negative emotion
739	more that just location, PixieEngine! RT @mention Google says the future is location, location, location: {link} #SXSW #CNN	Google	Negative emotion
745	iPhone, I know this #SXSW week will be tough on your already-dwindling battery, but so help me Jeebus if you keep correcting my curse words.	iPhone	Negative emotion

	tweet_text	product	emotion
756	Google to Launch Major New Social Network Called Circles (Updated) {link} *Not launched at #SXSW, but soon. Should I care?	Other Google product or service	Negative emotion
818	Google to launch product! Wait, no launch, but product exists. Wait, product does not exist! #sxsw {link}	Google	Negative emotion

- Rows 696, 713, 721, 746, 757, and 820 are definitely negative about products or brands.
- Rows 699, 700, and 707 are negative as well, but about apps, not really about the brand of phone they're running on.
- The only one I don't agree with is Row 740, which I don't think has any sentiment towards a brand (unless there was something in the removed link).

So I agreed with 9 out of 10 negative tweet labels in this sample; more than the positive labels.

In [159…
```
# let's get a sample of "no emotion"
df.loc[df['emotion']=="No emotion toward brand or product",
       ['tweet_text', 'product', 'emotion']][45:55]
```

Out[159…

	tweet_text	product	emotion
106	Attending "left brain search = Google, Right brain search = X" #Bettersearch -- talking about the future of search engines at #sxsw	NaN	No emotion toward brand or product
107	#HP opens "Mobile Park" & Content Incubator at #SXSW {link} #Apple constructs "pop-up" store {link}	NaN	No emotion toward brand or product
109	Kawasaki: "pagemaker saved Apple." Oh those were the days. #sxsw #jwtatl #enchantment	NaN	No emotion toward brand or product
112	Unboxing. #Apple #sxsw @mention Apple Store, SXSW {link}	NaN	No emotion toward brand or product
114	At #SXSW, #Apple schools the #marketing experts | SXSW - CNET Blogs {link}	NaN	No emotion toward brand or product
116	At #SXSW, #Apple schools the marketing experts {link}	NaN	No emotion toward brand or product
121	Headed to #Austin for #SXSW? Check out my map for newbies {link} @mention @mention , @mention @mention Enjoy!	NaN	No emotion toward brand or product
122	Funny how #Austin is trending but not #SXSW. Only a matter of minutes at this point (at least according to Twitter for iPhone).	NaN	No emotion toward brand or product
124	#sxsw #ux #ipad #uxdes remember to ultimately be aware of the audience your app is targeted towards. An unexpected experience can be good.	NaN	No emotion toward brand or product
128	#Google's #Mobile Future, and the Elusive 'Power of Here' - {link} (via @mention #eurorscg #sxsw #sxswi	NaN	No emotion toward brand or product

- Positive: row 106
- I agree with neutral: rows 107, 109
- Negative:

But wait, didn't we see the same tweet as row 109 in the positive sample? So that is definitely coded inconsistently.

Final Model Performance

The best performing models of the approaches I tried were the ones generated with randomly oversampled training data. Even though I used the `class_weight='balanced'`, I think providing more examples of the negative class was helpful in reducing misclassification of that class.

Binary Problem

The best binary model had a balanced accuracy rate of 75% on test data, so it was able to correctly classify tweets 75% of the time. However, this is an average: the model performed better predicting positive emotions than negative.

- About ~40% of negative tweets were mis-classified as positive, with ~60% of negative tweets being classified correctly
- Only about 10% of positive tweets were misclassified as negative, with about 90% of positive tweets being classified correctly.

Multi-class Problem

The best multi-class model had a balanced accuracy rate of ~62% on test data. The model was generally better at classifying tweets with No brand emotion or Positive brand emotion compared to tweets with Negative brand emotion, although the gap was not as wide as in the binary problem.

- For No emotion and Positive, the model made the correct classification about 65% of the time. About 25-28% of the time, it confused No emotion with Positive, or vice versa. It only misclassified No emotion or Positive as Negative ~8-10% of the time.
- For Negative, the model made the correct classification about 58% of the time, so a slightly lower rate than the other classes. ~20-22% of of the time, Negative tweets were confused for Positive or No emotion, about evenly split between them.

Overfitting and Generalization

Nearly all of the models were quite overfit to the training data, despite using regularization, and a quite short max_depth of 5 for the Random Forest.

I made multiple train-test-splits without setting a random seed to try to confirm it wasn't just a fluke in the way the data was split up one time. Although the results on test certainly varied depending on the split, the models always performed between 9 and 23% better on Test than Train. Average cross-validation scores for Train also differed from solo Train scores by 5-10%.

CONCLUSIONS & RECOMMENDATIONS

Provide your conclusions about the work you've done, including any limitations or next steps.

Questions to consider:

- What would you recommend the business do as a result of this work?
- What are some reasons why your analysis might not fully solve the business problem?
- What else could you do in the future to improve this project?

One big challenge with this dataset is how applicable it would be to other products, and also to different situations. I believe all of these tweets were tagged to the SWSW festival in Austin, so the content is specific to the activities and such that were there.

It's also pretty specific to Apple and Google products.

Future Improvements

- Try a more complex model or pre-trained vocabulary to see if it can do a better job at the multi-class problem. Even if it's less interpretable, perhaps it could help sift the tweets with emotions about brands or products versus those that have no emotion. The tweets with emotions could be fed into a more interpretable binary model to determine positive or negative, and the ngrams that contribute to both.
- Try separating out the emotions about apps from those about actual brands or products the brands control which is a more realistic use case.
- More examples of emotional tweets about Apple and Google would be needed to make a model that could generalize beyond the events at SXSW, or the products being released at that time.

In [ ]:

In [ ]: