Twitter Sentiment and Modeling

Overview

This project utilizes a dataset from CrowdFlower, analyzing and rating the sentiment of Twitter users regarding Apple and Google products by building an NLP model. Human raters rated the sentiment in over 9,000 Tweets as positive, negative, or neither.

Business Problem

Apple and Google want to gather information on the consensus of their products. They are looking at Twitter as a medium to gather that information. The task is to build a model that can rate the sentiment of a Tweet based on its content.

Data Understanding

The dataset used for this project is a csv file ("data.csv"), containing over 9,000 Tweets about Apple and Google products. Human raters rated the sentiment as positive, negative, or neither. The target column is the sentiment column.

Methods

This project uses descriptive analysis, exploratory data analysis, data visualization, natural language processing, and machine building. This provides key insights to optimizing the predictive ability of customers' satisfaction with brands and products.

Import Libraries

First thing we did was import the necessary libraries for analysis, visualization, preprocessing data, and building models, as well as ignore warnings.

In [1]:
```python
#import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
import numpy as np
import math
import seaborn as sns

import nltk
from nltk import FreqDist, ngrams, TweetTokenizer
from nltk.corpus import stopwords
from nltk.tokenize import RegexpTokenizer, word_tokenize
from nltk.stem import PorterStemmer

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')

import string

from wordcloud import WordCloud

from collections import Counter

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_sc
from sklearn.pipeline import Pipeline
import imblearn
from imblearn.pipeline import Pipeline
from imblearn.under_sampling import RandomUnderSampler

import warnings
from pandas.core.common import SettingWithCopyWarning

#Ignore feature warnings
warnings.filterwarnings("ignore", category=FutureWarning)
#Ignore copy warnings
warnings.filterwarnings("ignore", category=SettingWithCopyWarning)
```

```
[nltk_data] Downloading package punkt to /Users/Chris/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/Chris/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /Users/Chris/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Data Inspection

We proceeded to load the csv dataset, then look at the shape, size, column names and data types, as well as check for missing or duplicate entries.

In [2]:
```
#load the dataset, ensure the proper encoding is read
df = pd.read_csv('data.csv', encoding='latin1')
df.head()
```

Out[2]:

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_or_pro
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative em
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive em
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive em
3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Negative em
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Positive em

In [3]:
```
#change the name of the tweet, product, and sentiment columns
df = df.rename(columns={'tweet_text': 'tweet', 'emotion_in_tweet_is_di
df.head()
```

Out[3]:

	tweet	brand_or_product	sentiment
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative emotion
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive emotion
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive emotion
3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Negative emotion
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Positive emotion

In [4]:
```
#look at the different values for sentiment column
df['sentiment'].value_counts()
```

Out[4]:
```
No emotion toward brand or product    5389
Positive emotion                      2978
Negative emotion                       570
I can't tell                           156
Name: sentiment, dtype: int64
```

In [5]:
```
#look at the 'I can't tell' rows
df.loc[df['sentiment'] == "I can't tell"].head()
```

Out[5]:

	tweet	brand_or_product	sentiment
90	Thanks to @mention for publishing the news of ...	NaN	I can't tell
102	ÛÏ@mention &quot;Apple has opened a pop-up st...	NaN	I can't tell
237	Just what America needs. RT @mention Google to...	NaN	I can't tell
341	The queue at the Apple Store in Austin is FOUR...	NaN	I can't tell
368	Hope it's better than wave RT @mention Buzz is...	NaN	I can't tell

Drop the "I can't tell" rows

We decided to drop the rows labeled "I can't tell", as they would only serve to confuse the dataset, and didn't make up a significant portion of the dataset anyway.

In [6]:
```
#drop the I can't tell rows
mask = df['sentiment'] == "I can't tell"
df.drop(df[mask].index, inplace=True)
print(df['sentiment'].value_counts())
```

```
No emotion toward brand or product    5389
Positive emotion                      2978
Negative emotion                       570
Name: sentiment, dtype: int64
```

Change Sentiment Values

We decided to combine 'I can't tell' and 'No emotion toward brand or product' into the value 'Neutral', and change 'Positive emotion' and 'Negative emotion' to just 'Positive' and 'Negative'.

In [7]:
```
#change sentiment values
df['sentiment'] = df['sentiment'].replace({'No emotion toward brand or
                                          'Positive emotion': 'Positi
                                          'Negative emotion': 'Negati
print(df['sentiment'].value_counts())
```

```
Neutral     5389
Positive    2978
Negative     570
Name: sentiment, dtype: int64
```

In [8]:
```
#look at the different values for brand_or_product column
df['brand_or_product'].value_counts()
```

Out[8]:
```
iPad                               942
Apple                              659
iPad or iPhone App                 470
Google                             429
iPhone                             296
Other Google product or service    292
Android App                         81
Android                             78
Other Apple product or service      35
Name: brand_or_product, dtype: int64
```

In [9]:
```
#check information on each column
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8937 entries, 0 to 9092
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   tweet             8936 non-null   object
 1   brand_or_product  3282 non-null   object
 2   sentiment         8937 non-null   object
dtypes: object(3)
memory usage: 279.3+ KB
```

Duplicates

We checked for and found duplicate records, then proceeded to drop them.

In [10]:
```
#check for duplicates
df[df.duplicated()]
```

Out[10]:

	tweet	brand_or_product	sentiment
468	Before It Even Begins, Apple Wins #SXSW {link}	Apple	Positive
776	Google to Launch Major New Social Network Call...	NaN	Neutral
2232	Marissa Mayer: Google Will Connect the Digital...	NaN	Neutral
2559	Counting down the days to #sxsw plus strong Ca...	Apple	Positive
3950	Really enjoying the changes in Gowalla 3.0 for...	Android App	Positive
3962	#SXSW is just starting, #CTIA is around the co...	Android	Positive
4897	Oh. My. God. The #SXSW app for iPad is pure, u...	iPad or iPhone App	Positive
5338	RT @mention ÷¼ GO BEYOND BORDERS! ÷_ {link} ...	NaN	Neutral
5341	RT @mention ÷¼ Happy Woman's Day! Make love, ...	NaN	Neutral
5881	RT @mention Google to Launch Major New Social ...	NaN	Neutral
5882	RT @mention Google to Launch Major New Social ...	NaN	Neutral
5883	RT @mention Google to Launch Major New Social ...	NaN	Neutral
5884	RT @mention Google to Launch Major New Social ...	NaN	Neutral
5885	RT @mention Google to Launch Major New Social ...	NaN	Neutral
6296	RT @mention Marissa Mayer: Google Will Connect...	Google	Positive
6297	RT @mention Marissa Mayer: Google Will Connect...	NaN	Neutral
6298	RT @mention Marissa Mayer: Google Will Connect...	Google	Positive
6299	RT @mention Marissa Mayer: Google Will Connect...	NaN	Neutral
6300	RT @mention Marissa Mayer: Google Will Connect...	NaN	Neutral
6546	RT @mention RT @mention Google to Launch Major...	NaN	Neutral
8483	I just noticed DST is coming this weekend. How...	iPhone	Negative
8747	Need to buy an iPad2 while I'm in Austin at #s...	iPad	Positive

In [11]:
```
#check the number of duplicates
print(len(df[df.duplicated()]))
```

```
22
```

In [12]:
```
#drop duplicates
df.drop_duplicates(inplace=True)
df[df.duplicated()]
```

Out[12]:

tweet	brand_or_product	sentiment

Missing Values

We checked for missing values and were missing 1 value for the tweet column and almost 6,000 values for the brand_or_product column.

In [13]:
```
#look at the row with the missing value for the 'tweet' column
df.loc[df['tweet'].isnull()]
```

Out[13]:

	tweet	brand_or_product	sentiment
6	NaN	NaN	Neutral

Drop Missing Tweet

Since there is nothing useful provided in the entire row that's the sole missing tweet, we just dropped the row.

In [14]:
```
#drop missing tweet row
df.dropna(subset=['tweet'], inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8914 entries, 0 to 9092
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   tweet             8914 non-null   object
 1   brand_or_product  3273 non-null   object
 2   sentiment         8914 non-null   object
dtypes: object(3)
memory usage: 278.6+ KB
```

Now we took a look at the missing brand/product rows.

In [15]:
```
#look at 20 rows of missing brand/product values
df.loc[df['brand_or_product'].isnull()].head(20)
```

Out[15]:

	tweet	brand_or_product	sentiment
5	@teachntech00 New iPad Apps For #SpeechTherapy...	NaN	Neutral
16	Holler Gram for iPad on the iTunes App Store -...	NaN	Neutral
32	Attn: All #SXSW frineds, @mention Register fo...	NaN	Neutral
33	Anyone at #sxsw want to sell their old iPad?	NaN	Neutral
34	Anyone at #SXSW who bought the new iPad want ...	NaN	Neutral
35	At #sxsw. Oooh. RT @mention Google to Launch ...	NaN	Neutral
37	SPIN Play - a new concept in music discovery f...	NaN	Neutral
39	VatorNews - Google And Apple Force Print Media...	NaN	Neutral
41	HootSuite - HootSuite Mobile for #SXSW ~ Updat...	NaN	Neutral
42	Hey #SXSW - How long do you think it takes us ...	NaN	Neutral
43	Mashable! - The iPad 2 Takes Over SXSW [VIDEO]...	NaN	Neutral
44	For I-Pad ?RT @mention New #UberSocial for #iP...	NaN	Neutral
46	Hand-Held □Û÷Hobo□Ûª: Drafthouse launches □Û÷H...	NaN	Positive
48	Orly....? □ÛÏ@mention Google set to launch new...	NaN	Neutral
50	Khoi Vinh (@mention says Conde Nast's headlong...	NaN	Neutral
51	□ÛÏ@mention {link} &lt;-- HELP ME FORWARD THIS...	NaN	Neutral
52	□÷¼ WHAT? □÷_ {link} □ã_ #edchat #musedchat #s...	NaN	Neutral
53	.@mention @mention on the location-based 'fast...	NaN	Neutral
54	□ÛÏ@mention @mention #Google Will Connect the ...	NaN	Neutral
56	{link} RT @mention &quot;Google before you twe...	NaN	Neutral

We looked for any correlation or pattern between missing brands and the sentiment.

In [16]:
```
#check missing brand/product rows that have a sentiment other than Neu
df.loc[(df['brand_or_product'].isnull()) & (df['sentiment'] != 'Neutra
```

Out[16]:

	tweet	brand_or_product	sentiment
46	Hand-Held ÛÏ÷HoboÛÏª: Drafthouse launches ÛÏ÷H...	NaN	Positive
64	Again? RT @mention Line at the Apple store is ...	NaN	Negative
68	Boooo! RT @mention Flipboard is developing an ...	NaN	Negative
103	Know that &quot;dataviz&quot; translates to &q...	NaN	Negative
112	Spark for #android is up for a #teamandroid aw...	NaN	Positive
...	...	...	...
9011	apparently the line to get an iPad at the #sxs...	NaN	Positive
9043	Hey is anyone doing #sxsw signing up for the g...	NaN	Negative
9049	@mention you can buy my used iPad and I'll pic...	NaN	Positive
9052	@mention You could buy a new iPad 2 tmrw at th...	NaN	Positive
9054	Guys, if you ever plan on attending #SXSW, you...	NaN	Positive

357 rows × 3 columns

In [17]:
```
#print the number of missing brand/product rows that have a sentiment
print("Number of rows with a sentiment other than Neutral: ", len(df.l
print("Number of rows with Neutral as the sentiment: ", len(df.loc[(df
```

```
Number of rows with a sentiment other than Neutral:  357
Number of rows with Neutral as the sentiment:  5284
```

Modify Null Brand/Product Rows

With so many entries missing a value for the brand/product, and having a non-neutral sentiment, we proceed to write a function to fill in the brand/product if it contained one of our brand/product values in the tweet, then applied this function to the dataset.

In [18]:
```
#function to change the brand/product column based on inclusion of one
def get_brand_or_product(tweet):
    tweet = tweet.lower()
    keywords = ['apple', 'google', 'iphone', 'ipad', 'android']
    count = 0
    brand_product = None

    for keyword in keywords:
        if keyword in tweet:
            count += 1
            brand_product = keyword
    #leave the column blank if more than one of the brand/product valu
    if count > 1:
        brand_product = None

    return brand_product
```

In [19]:
```
#apply the function to the null rows in the dataset
mask = df['brand_or_product'].isnull()
df.loc[mask, 'brand_or_product'] = df.loc[mask, 'tweet'].apply(get_bra
print(len(df.loc[df['brand_or_product'].isnull()]))
```

```
1492
```

In [20]:
```
#check the info on each column again
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8914 entries, 0 to 9092
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   tweet             8914 non-null   object
 1   brand_or_product  7422 non-null   object
 2   sentiment         8914 non-null   object
dtypes: object(3)
memory usage: 278.6+ KB
```

In [21]:
```
#look at 20 null rows again
df.loc[df['brand_or_product'].isnull()].head(20)
```

Out[21]:

	tweet	brand_or_product	sentiment
39	VatorNews - Google And Apple Force Print Media...	None	Neutral
41	HootSuite - HootSuite Mobile for #SXSW ~ Updat...	None	Neutral
51	□ÛÏ@mention {link} &lt;-- HELP ME FORWARD THIS...	None	Neutral
52	□÷¼ WHAT? □÷_ {link} □ã_ #edchat #musedchat #s...	None	Neutral
53	.@mention @mention on the location-based 'fast...	None	Neutral
66	At #sxsw? @mention / @mention wanna buy you a ...	None	Neutral
68	Boooo! RT @mention Flipboard is developing an ...	None	Negative
71	Chilcott: @mention #SXSW stand talking with Bl...	None	Neutral
73	Gowalla's @mention promises to launch Foursqua...	None	Neutral
77	I worship @mention {link} #SXSW	None	Neutral
79	Launching @mention #SxSW? RT @mention @mention...	None	Neutral
82	Nice! RT @mention Apple opening popup store f...	None	Neutral
85	Stay tune @mention showcase #H4ckers {link} #SXSW	None	Neutral
86	Thank you @mention @mention for the #touchings...	None	Neutral
87	Thank you @mention for an awesome #sxsw party!...	None	Neutral
88	Thanks RT @mention If you're trying to contact...	None	Neutral
91	Thanks to @mention for publishing the news of ...	None	Neutral
93	Wonder if @mention &amp; @mention will be in t...	None	Neutral
94	Wonder if @mention is putting tips from the @m...	None	Neutral
97	Yes!!! RT @mention hey @mention , i've got ano...	None	Neutral

Placeholder

We decided to put a placeholder of 'Unknown' for the rest of the null rows.

In [22]:
```
#fill missing rows with 'Unknown'
df['brand_or_product'].fillna('Unknown', inplace=True)
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8914 entries, 0 to 9092
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   tweet             8914 non-null   object
 1   brand_or_product  8914 non-null   object
 2   sentiment         8914 non-null   object
dtypes: object(3)
memory usage: 278.6+ KB
None
```

In [23]:
```
#check values for brand_or_product again
df['brand_or_product'].value_counts()
```

Out[23]:
```
google                             1639
Unknown                            1492
ipad                                946
iPad                                941
iphone                              684
apple                               671
Apple                               657
iPad or iPhone App                  469
Google                              427
iPhone                              295
Other Google product or service     292
android                             209
Android App                          80
Android                              77
Other Apple product or service       35
Name: brand_or_product, dtype: int64
```

After our work with the brand_or_product column, we proceeded to limit the values in that column for consistency with capitalization.

In [24]:
```
#change brand_or_product values
df['brand_or_product'] = df['brand_or_product'].replace({'google': 'Go
                                                         'ipad': 'iPad',
                                                         'iphone': 'iPhone',
                                                          'apple': 'Apple',
                                                          'android': 'Android'})
print(df['brand_or_product'].value_counts())
```

```
Google                             2066
iPad                               1887
Unknown                            1492
Apple                              1328
iPhone                              979
iPad or iPhone App                  469
Other Google product or service     292
Android                             286
Android App                          80
Other Apple product or service       35
Name: brand_or_product, dtype: int64
```

Data Cleaning

We performed standard actions such as standardizing and tokenizing the data.

Standardizing Case

We explored some tweets and decided to lowercase all text.

In [25]:
```
#look at examples of tweets to inspect for spelling
print(df["tweet"].to_list()[:10])
```

```
['.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austi
n, it was dead!  I need to upgrade. Plugin stations at #SXSW.', "@jes
sedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likel
y appreciate for its design. Also, they're giving free Ts at #SXSW",
'@swonderlin Can not wait for #iPad 2 also. They should sale them dow
n at #SXSW.', "@sxsw I hope this year's festival isn't as crashy as t
his year's iPhone app. #sxsw", "@sxtxstate great stuff on Fri #SXSW:
Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) &amp; M
att Mullenweg (Wordpress)", '@teachntech00 New iPad Apps For #SpeechT
herapy And Communication Are Showcased At The #SXSW Conference htt
p://ht.ly/49n4M (http://ht.ly/49n4M) #iear #edchat #asd', '#SXSW is j
ust 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', 'Beauti
fully smart and simple idea RT @madebymany @thenextweb wrote about ou
r #hollergram iPad app for #sxsw! http://bit.ly/ieaVOB', (http://bit.
ly/ieaVOB',) 'Counting down the days to #sxsw plus strong Canadian do
llar means stock up on Apple gear', 'Excited to meet the @samsungmobi
leus at #sxsw so I can show them my Sprint Galaxy S still running And
roid 2.1.   #fail']
```

Given that we have instances of SXSW and sxsw that we want to treat as the same, and there presumably may be instances of different capitalizations of terms like "Apple" and "iPhone", we proceed to lowercase all tweets in our dataset.

In [26]:
```
# Transform tweets to lowercase
df["tweet"] = df["tweet"].str.lower()

#print example
print(df.iloc[50]["tweet"])
```

```
ûï@mention {link} &lt;-- help me forward this doc to all anonymous a
ccounts, techies,&amp; ppl who can help us jam #libya #sxsw
```

Tokenize the Full Dataset

We proceeded to create a tokenizer pattern and test it on a sample of rows

```
In [27]: #create a tokenizer pattern to avoid unnecessary separate treatment of
         basic_token_pattern = r"(?u)\b\w\w+\b"

         tokenizer = RegexpTokenizer(basic_token_pattern)

         tweets = df.loc[:5, "tweet"].tolist()  # Convert the slice to a list c
         tokenized_tweets = [tokenizer.tokenize(tweet) for tweet in tweets[:10]
         print(tokenized_tweets)
```

```
[['wesley83', 'have', '3g', 'iphone', 'after', 'hrs', 'tweeting', 'a
t', 'rise_austin', 'it', 'was', 'dead', 'need', 'to', 'upgrade', 'plu
gin', 'stations', 'at', 'sxsw'], ['jessedee', 'know', 'about', 'fluda
pp', 'awesome', 'ipad', 'iphone', 'app', 'that', 'you', 'll', 'likel
y', 'appreciate', 'for', 'its', 'design', 'also', 'they', 're', 'givi
ng', 'free', 'ts', 'at', 'sxsw'], ['swonderlin', 'can', 'not', 'wai
t', 'for', 'ipad', 'also', 'they', 'should', 'sale', 'them', 'down',
'at', 'sxsw'], ['sxsw', 'hope', 'this', 'year', 'festival', 'isn', 'a
s', 'crashy', 'as', 'this', 'year', 'iphone', 'app', 'sxsw'], ['sxtxs
tate', 'great', 'stuff', 'on', 'fri', 'sxsw', 'marissa', 'mayer', 'go
ogle', 'tim', 'reilly', 'tech', 'books', 'conferences', 'amp', 'mat
t', 'mullenweg', 'wordpress'], ['teachntech00', 'new', 'ipad', 'app
s', 'for', 'speechtherapy', 'and', 'communication', 'are', 'showcase
d', 'at', 'the', 'sxsw', 'conference', 'http', 'ht', 'ly', '49n4m', '
iear', 'edchat', 'asd']]
```

We then applied the pattern to the entire dataframe, creating a new column to display the results.

In [28]:
```
# Create new column with tokenized data
df["tweet_tokenized"] = df["tweet"].apply(tokenizer.tokenize)
# Display full text
df.head().style.set_properties(**{'text-align': 'left'})
```

Out[28]:

	tweet	brand_or_product	sentiment	tweet_tokenized
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	['wesley83', 'have', '3g', 'iphone', 'after', 'hrs', 'tweeting', 'at', 'rise_austin', 'it', 'was', 'dead', '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	iPad or iPhone App	Positive	['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	['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	iPad or iPhone App	Negative	['sxsw', 'hope', 'this', 'year', 'festival', 'isn', 'as', 'crashy', 'as', 'this', 'year', 'iphone', 'app', 'sxsw']
4	@sxtxstate great stuff on fri #sxsw: marissa mayer (google), tim o'reilly (tech books/conferences) & matt mullenweg (wordpress)	Google	Positive	['sxtxstate', 'great', 'stuff', 'on', 'fri', 'sxsw', 'marissa', 'mayer', 'google', 'tim', 'reilly', 'tech', 'books', 'conferences', 'amp', 'matt', 'mullenweg', 'wordpress']

Stop Word Removal

First we got all the english stop words and stored them in a variable.

In [29]:
```
#create a stop words list and add all english words, plus punctuation
stopwords_list = stopwords.words('english')
stopwords_list += list(string.punctuation)
stopwords_list += ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
stopwords_list += ['"', '...', "''", "'", '``', '']

#store stop words in variable
tweet_words_stopped = [word for row in df['tweet_tokenized'] for word
```

Then we created a frequency distribution to see if removing stop words helped.

In [30]:
```
#create a frequency distribution of the stop words list
tweet_stopped_freqdist = FreqDist(tweet_words_stopped)

#display the 50 most common
print(tweet_stopped_freqdist.most_common(50))
```

```
[('sxsw', 9444), ('mention', 7006), ('link', 4249), ('rt', 2919), ('g
oogle', 2602), ('ipad', 2472), ('apple', 2294), ('quot', 1657), ('iph
one', 1551), ('store', 1463), ('new', 1075), ('austin', 955), ('amp',
827), ('app', 819), ('circles', 649), ('social', 648), ('launch', 64
0), ('android', 590), ('pop', 586), ('today', 569), ('ipad2', 459),
('network', 452), ('via', 428), ('line', 401), ('get', 392), ('free',
390), ('party', 349), ('called', 347), ('mobile', 345), ('sxswi', 33
8), ('one', 309), ('major', 296), ('like', 284), ('time', 270), ('tem
porary', 264), ('opening', 256), ('check', 254), ('possibly', 233),
('day', 230), ('people', 226), ('downtown', 225), ('apps', 222), ('gr
eat', 221), ('see', 221), ('maps', 217), ('open', 214), ('going', 21
3), ('mayer', 212), ('popup', 210), ('go', 205)]
```

Next we created a function to remove the stop words from our tokenized tweets.

In [31]:
```
# Function to remove stopwords and additional words
def remove_stopwords(row):
    tokens = row['tweet_tokenized']
    filtered_tokens = [word for word in tokens if word not in stopword
    return filtered_tokens

# Apply the function to remove stopwords and additional words
df['stop_tweet_tokenized'] = df.apply(lambda row: remove_stopwords(row

# Display
df.head().style.set_properties(**{'text-align': 'left'})
```

Out[31]:

tweet	brand_or_product	sentiment	tweet_tokenized	stop_tweet_tokenized
			['wesley83',	

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	'have', '3g', 'iphone', 'after', 'hrs', 'tweeting', 'at', 'rise_austin', 'it', 'was', 'dead', 'need', 'to', 'upgrade', 'plugin', 'stations', 'at', 'sxsw']	['wesley83', '3g', 'iphone', 'hrs', 'tweeting', 'rise_austin', 'dead', 'need', 'upgrade', 'plugin', 'stations', '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	iPad or iPhone App	Positive	['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', 'fludapp', 'awesome', 'ipad', 'iphone', 'app', 'likely', 'appreciate', 'design', 'also', 'giving', 'free', 'ts', 'sxsw']
2	@swonderlin can not wait for #ipad 2 also. they should sale them down at #sxsw.	iPad	Positive	['swonderlin', 'can', 'not', 'wait', 'for', 'ipad', 'also', 'they', 'should', 'sale', 'them', 'down', 'at', 'sxsw']	['swonderlin', 'wait', 'ipad', 'also', 'sale', '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	['sxsw', 'hope', 'this', 'year', 'festival', 'isn', 'as', 'crashy', 'as', 'this', 'year', 'iphone', 'app', 'sxsw']	['sxsw', 'hope', 'year', 'festival', 'crashy', 'year', 'iphone', 'app', 'sxsw']
4	@sxtxstate great stuff on fri #sxsw: marissa mayer (google), tim o'reilly (tech books/conferences) & matt mullenweg (wordpress)	Google	Positive	['sxtxstate', 'great', 'stuff', 'on', 'fri', 'sxsw', 'marissa', 'mayer', 'google', 'tim', 'reilly', 'tech', 'books', 'conferences', 'amp', 'matt', 'mullenweg', 'wordpress']	['sxtxstate', 'great', 'stuff', 'fri', 'sxsw', 'marissa', 'mayer', 'google', 'tim', 'reilly', 'tech', 'books', 'conferences', 'amp', 'matt', 'mullenweg', 'wordpress']

Stemming The Tokenized Text

Next thing we did was create a stemming function to ensure we don't lose important text when we remove stop words.

In [32]:
```
#instantiate a PorterStemmer function
```

```
stemmer = PorterStemmer()

#apply the function to the stop_tweet_tokenized column
df['stop_tweet_stemmed'] = df['stop_tweet_tokenized'].apply(lambda x:

# Display full text
df.head().style.set_properties(**{'text-align': 'left'})
```

Out[32]:

	tweet	brand_or_product	sentiment	tweet_tokenized	stop_tweet_tokenized	sto
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	['wesley83', 'have', '3g', 'iphone', 'after', 'hrs', 'tweeting', 'at', 'rise_austin', 'it', 'was', 'dead', 'need', 'to', 'upgrade', 'plugin', 'stations', 'at', 'sxsw']	['wesley83', '3g', 'iphone', 'hrs', 'tweeting', 'rise_austin', 'dead', 'need', 'upgrade', 'plugin', 'stations', 'sxsw']	['we 'iph 'rise 'nee 'plu 'sxs
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	['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', 'fludapp', 'awesome', 'ipad', 'iphone', 'app', 'likely', 'appreciate', 'design', 'also', 'giving', 'free', 'ts', 'sxsw']	['jes 'fluc 'ipa 'like 'des 'free
2	@swonderlin can not wait for #ipad 2 also. they should sale them down at #sxsw.	iPad	Positive	['swonderlin', 'can', 'not', 'wait', 'for', 'ipad', 'also', 'they', 'should', 'sale', 'them', 'down', 'at', 'sxsw']	['swonderlin', 'wait', 'ipad', 'also', 'sale', 'sxsw']	['sw 'ipa 'sxs
3	@sxsw i hope this year's festival isn't as crashy as this year's iphone app. #sxsw	iPad or iPhone App	Negative	['sxsw', 'hope', 'this', 'year', 'festival', 'isn', 'as', 'crashy', 'as', 'this', 'year', 'iphone', 'app', 'sxsw']	['sxsw', 'hope', 'year', 'festival', 'crashy', 'year', 'iphone', 'app', 'sxsw']	['sx 'fes 'iph
4	@sxtxstate great stuff on fri #sxsw: marissa mayer (google), tim o'reilly (tech	Google	Positive	['sxtxstate', 'great', 'stuff', 'on', 'fri', 'sxsw', 'marissa', 'mayer', 'google', 'tim', 'reilly',	['sxtxstate', 'great', 'stuff', 'fri', 'sxsw', 'marissa', 'mayer', 'google', 'tim', 'reilly', 'tech', 'books',	['sx 'stu 'ma 'goo 'tec

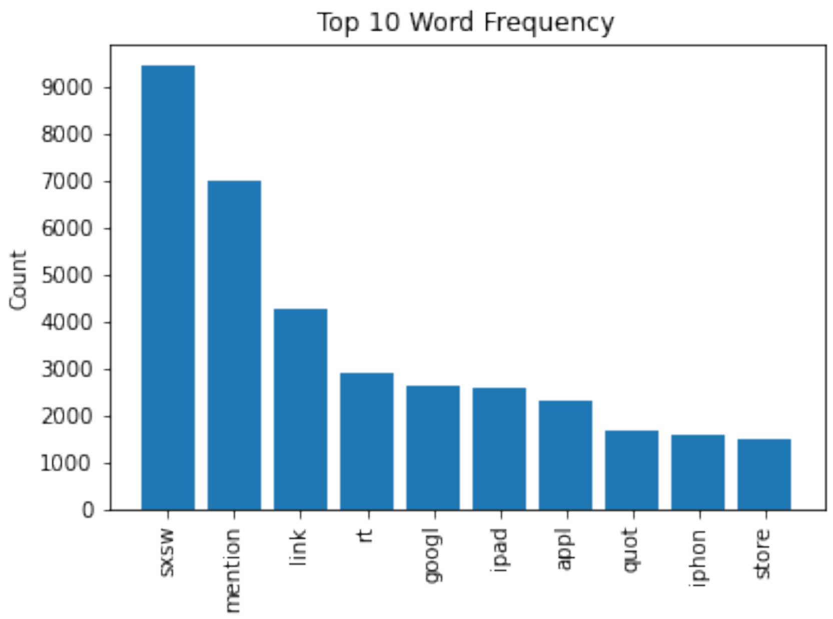
books/conferences) & matt mullenweg (wordpress)	'tech', 'books', 'conferences', 'amp', 'matt', 'mullenweg', 'wordpress']	'conferences', 'amp', 'matt', 'mullenweg', 'wordpress']	'am 'mu 'wo

Exploratory Data Analysis: Frequency Distributions

In this section, we looked at the frequency of words from the tweets in our dataset.

In [33]:
```
#write a function for visualizing the top 10 most frequent words
def visualize_top_10(freq_dist, title):

    # Extract data for plotting
    top_10 = list(zip(*freq_dist.most_common(10)))
    tokens = top_10[0]
    counts = top_10[1]

    # Set up plot and plot data
    fig, ax = plt.subplots()
    ax.bar(tokens, counts)

    # Customize plot appearance
    ax.set_title(title)
    ax.set_ylabel("Count")
    ax.yaxis.set_major_locator(MaxNLocator(integer=True))
    ax.tick_params(axis="x", rotation=90)
```

In [34]:
```
# Create a frequency distribution for X_train
df_freq_dist = FreqDist(df["stop_tweet_stemmed"].explode())

# Plot the top 10 tokens
visualize_top_10(df_freq_dist, "Top 10 Word Frequency")
```



We saw that there continues to be stop words in our top 10 frequency distribution that need to be removed.

More Word Inspection and Removal

We continue to look at the top frequent words and inspect their context, such as mention and link, to see if they don't serve us much purpose and need to be removed.

In [35]:
```
# Filter rows that contain 'sxsw' in 'stop_tweet_stemmed' column
sxsw_filtered_df = df[df['stop_tweet_stemmed'].apply(lambda x: 'sxsw'

# Set the column width option to display the full content
pd.set_option('display.max_colwidth', None)

# Print a sample of rows from the filtered dataframe
sxsw_sample_rows = sxsw_filtered_df['stop_tweet_stemmed'].sample(n=10)
print(sxsw_sample_rows)
```

```
2097                                                              [go,
bar, get, free, drink, iphon, doesdroid, sxsw]
3923                                                            [amp,
bing, amp, googl, let, game, begin, qagb, sxsw]
8271
[googl, hot, pot, whattt, pot, sxsw]
4324                  [hehe, rt, ûï, mention, march, 11, austin, tx,
peopl, line, sxsw, registr, appl, store, mobil]
4177                              [ipad2, deliveri, pop, mention, st
ore, mention, quit, possibl, sxsw, sxswi, link]
4415                             [mike, tyson, come, phone, near, li
nk, iphon, sxsw, videogam, tyson, art, cartoon]
564                                                     [attend, goog
l, keynot, see, googl, map, mobil, featur, sxsw]
2310    [awkward, jc, penney, question, ask, marissa, mayer, appar, j
c, penney, locat, rout, maci, googl, map, sxsw]
5515              [rt, mention, sxsw, get, soundcloud, iphon, app, li
nk, start, record, use, 4sq, geotag, map, link]
2839
[pick, ipad, back, sxsw, link]
Name: stop_tweet_stemmed, dtype: object
```

In [36]:
```
#add sxsw to stop words
stopwords_list.append('sxsw')
stopwords_list.append('#sxsw')
```

In [37]:
```
# Filter rows that contain 'mention' in 'tweet_stemmed' column
mention_filtered_df = df[df['stop_tweet_stemmed'].apply(lambda x: 'mer

# Print a sample of rows from the filtered dataframe
mention_sample_rows = mention_filtered_df['stop_tweet_stemmed'].sample
print(mention_sample_rows)
```

```
1567                                                      [mention, ment
ion, appl, gotta, take, advantag, hipster, head, sxsw, somehow]
6929                                               [rt, mention, fli, sx
sw, want, mention, free, mile, dm, shoot, code, current, iphon]
2546                                     [hope, jinx, mention, nice, ment
ion, iphon, app, behav, today, crash, yesterday, ridicul, sxsw]
5167                       [rt, mention, quot, appl, come, cool, tech
nolog, one, ever, heard, go, confer, quot, sxsw, pseudoretweet]
5209                                          [rt, mention, appl, open, p
opup, shop, downtown, austin, sxsw, link, rt, mention, mention]
4837                                          [excit, part, mention, twit
ter, famili, stop, pepsico, playground, sxsw, learn, win, ipad]
6603    [rt, mention, rt, mention, rt, mention, googl, launch, major,
new, social, network, call, circl, possibl, today, link, sxsw]
6023                                [rt, mention, hmm, sxsw, com, inter
act, live, stream, ipad, mobil, compat, mayb, next, year, sxsw]
1659                                         [mention, amp, finish, mad,
dash, complet, ipad, format, web, app, client, show, sxsw, fun]
4719                                                      [wanna, know,
rt, mention, one, produc, go, sxsw, hope, free, iphon, app, dl]
Name: stop_tweet_stemmed, dtype: object
```

In [38]:
```
#add mention to stop words
stopwords_list.append('mention')
```

In [39]:
```
# Filter rows that contain 'link' in 'tweet_stemmed' column
link_filtered_df = df[df['stop_tweet_stemmed'].apply(lambda x: 'link'

# Print a sample of rows from the filtered dataframe
link_sample_rows = link_filtered_df['stop_tweet_stemmed'].sample(n=10)
print(link_sample_rows)
```

```
5591    [rt, mention, cameron, sinclair, mention, spearhead, japan, d
isast, relief, sxsw, via, twitter, amp, iphon, link, retweet]
8668                                            [googl, launch, secre
t, new, social, network, call, quot, circl, quot, link, sxsw]
4983                                [team, android, parti, 13, 10, show,
us, mention, app, mobil, enter, win, free, nexu, link, sxsw]
4376
[nope, seem, googl, circl, launch, today, link, sxsw]
2066                                                             [sxs
w, apptast, link, android, app, develop, io, iphon, smartphon]
1801          [mention, amp, mention, vs, mention, amp, quot, groupo
n, live, social, type, quot, reward, link, battl, sxsw, begin]
4381                                                            [quot,
commun, place, web, friend, app, quot, link, sxsw, grauniad]
5169                [rt, mention, quot, appl, like, pay, appl, like,
quot, barri, diller, sxsw, mention, acc, ballroom, pic, link]
6985           [rt, mention, ye, updat, iphon, app, song, info, menti
on, 24, stream, other, also, live, video, stream, sxsw, link]
731                                         [mention, bigger, ipho
n, smaller, pc, good, big, event, like, sxsw, meet, day, link]
Name: stop_tweet_stemmed, dtype: object
```

In [40]:
```
#add link to stop words
stopwords_list.append('link')
```

In [41]:
```
# Filter rows that contain 'rt' in 'tweet_stemmed' column
rt_filtered_df = df[df['stop_tweet_stemmed'].apply(lambda x: 'rt' in x

# Print a sample of rows from the filtered dataframe
rt_sample_rows = rt_filtered_df['stop_tweet_stemmed'].sample(n=10)
print(rt_sample_rows)
```

```
6344                                                               [rt, men
tion, new, sxsw, rule, oo, ahe, new, ipad, get, big, deal, everybodi,
one]
5713                   [rt, mention, fedex, truck, keep, arriv, tv, c
rew, interview, peopl, line, first, ipad2, sxsw, popup, appl, store,
link]
5365                                                                      [
rt, mention, browserwar, panel, without, appl, like, sxsw, without, p
arti]
3965                                                    [great, link, gt, gt,
rt, mention, link, ã_, edchat, musedchat, sxsw, sxswi, classic, newtw
itt]
5596
[rt, mention, sxsw, download, free, music, mix, itun, link, cc, menti
on]
6305                                              [rt, mention, mayer, make,
clear, googl, go, straight, foursquar, amp, gowalla, googl, hotpot, s
xsw]
5968                                                     [rt, mention, head,
link, 1pm, cst, today, win, vip, access, acoust, solo, set, sxsw, ton
ight]
4194                                            [suck, rt, mention, rt, ment
ion, googl, preview, major, new, social, servic, circl, sxsw, today,
link]
7914                                           [tweet, regist, exclus, pass,
event, parti, ipad, sxsw, quot, give, liberti, free, sxswpass, pleas,
rt]
6552    [rt, mention, rt, mention, googl, launch, major, new, social,
network, call, circl, possibl, today, mention, sxsw, link, via, menti
onw]
Name: stop_tweet_stemmed, dtype: object
```

In [42]:
```
#add rt to stop words
stopwords_list.append('rt')
```

In [43]:
```
# Filter rows that contain 'quot' in 'tweet_stemmed' column
quot_filtered_df = df[df['stop_tweet_stemmed'].apply(lambda x: 'quot'

# Print a sample of rows from the filtered dataframe
quot_sample_rows = quot_filtered_df['stop_tweet_stemmed'].sample(n=10)
print(quot_sample_rows)
```

```
6279                                                          [rt, me
ntion, love, mention, sxsw, quot, appl, come, cool, technolog, one, e
ver, heard, link]
5175    [rt, mention, quot, googl, quot, product, gatekeep, quot, mar
issa, mayer, locat, base, quot, fast, fun, futur, quot, link, ht, men
tion, sxsw, quot]
3457                                                                 
[hear, quot, design, ipad, interfac, new, navig, schema, quot, sxsw,
link, uxd]
4777                                        [quot, mention, hoot,
new, blog, post, hootsuit, mobil, sxsw, updat, iphon, bberri, androi
d, link, mention]
8102                                                                 
[quot, stay, aliv, indi, iphon, game, develop, surviv, quot, sxsw]
6962                    [rt, mention, woman, lobbi, quot, websit, cal
l, like, stupid, iphon, speller, ppl, take, pic, funni, autocorrect,
word, quot, sxsw]
7134                                          [nyt, app, ipad, quot,
amaz, way, serv, readership, quot, quot, market, opportun, ignor, quo
t, sxsw, newsapp]
4257                                                             [kin
gdom, way, filter, tweet, includ, word, quot, unlock, quot, twitter,
iphon, app, sxsw]
2905                                          [sxsw, attende, trade,
quot, happi, hour, quot, appi, hour, wait, line, ipad2, video, link,
sheer, mad, love]
3217                                   [best, thing, abt, mention, sx
sw, bad, ass, brunch, patio, amp, plenti, free, park, 10, min, quot,
mess, quot, link]
Name: stop_tweet_stemmed, dtype: object
```

In [44]:
```
#add quot to stop words
stopwords_list.append('quot')
```

Next we added a column to the dataframe that removes the additional stop words.

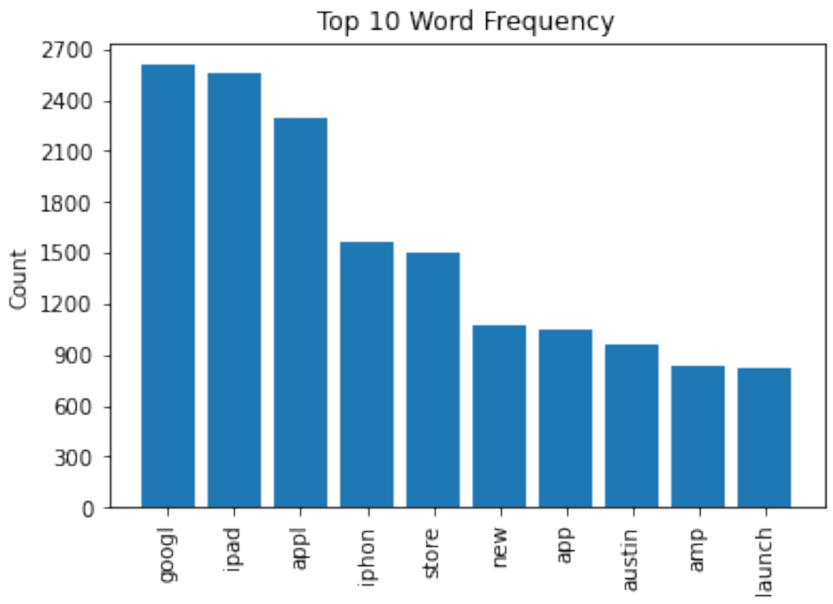
In [45]:
```
# Define a function to remove stopwords from text
def remove_extra_stopwords(text):
    filtered_words = [word for word in text if word not in stopwords_l
    return filtered_words
```

```
# Apply the remove_stopwords function to the 'stop_tweet_stemmed' colu
df['stop_tweet_stemmed'] = df['stop_tweet_stemmed'].apply(lambda x: re

# Display the updated DataFrame
df.head()
```

Out[45]:

	tweet	brand_or_product	sentiment	tweet_tokenized	stop_tweet_tokenized	s
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	[wesley83, have, 3g, iphone, after, hrs, tweeting, at, rise_austin, it, was, dead, need, to, upgrade, plugin, stations, at, sxsw]	[wesley83, 3g, iphone, hrs, tweeting, rise_austin, dead, need, upgrade, plugin, stations, 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	iPad or iPhone App	Positive	[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, fludapp, awesome, ipad, iphone, app, likely, appreciate, design, also, giving, free, ts, sxsw]	
2	@swonderlin can not wait for #ipad 2 also. they should sale them down at #sxsw.	iPad	Positive	[swonderlin, can, not, wait, for, ipad, also, they, should, sale, them, down, at, sxsw]	[swonderlin, wait, ipad, also, sale, 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	[sxsw, hope, this, year, festival, isn, as, crashy, as, this, year, iphone, app, sxsw]	[sxsw, hope, year, festival, crashy, year, iphone, app, sxsw]	
4	@sxtxstate great stuff on fri #sxsw: marissa mayer (google), tim o'reilly (tech books/conferences) &amp; matt mullenweg (wordpress)	Google	Positive	[sxtxstate, great, stuff, on, fri, sxsw, marissa, mayer, google, tim, reilly, tech, books, conferences, amp, matt, mullenweg, wordpress]	[sxtxstate, great, stuff, fri, sxsw, marissa, mayer, google, tim, reilly, tech, books, conferences, amp, matt, mullenweg, wordpress]	

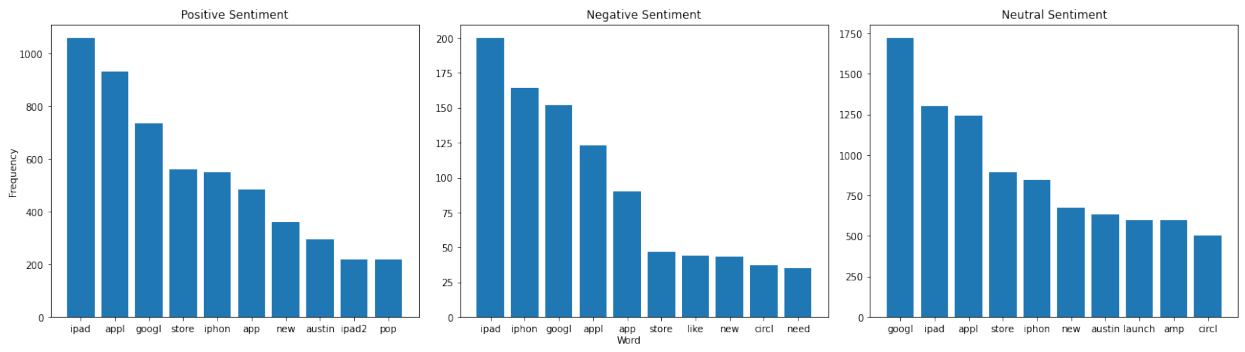
In [46]:
```
#update variable of cleaned text
tweet_words_stopped = [word for row in df['stop_tweet_stemmed'] for wo

#create a frequency distribution of the updated stopped words list
tweet_stopped_freqdist = FreqDist(tweet_words_stopped)

# Plot the top 10 updated tokens
visualize_top_10(tweet_stopped_freqdist, "Top 10 Word Frequency")
```



Now that the most frequent words seem to better apply to the text of tweets that provide context, the next thing we do is visually inspect these words categorized by sentiment.

In [47]:
```
#list of cleaned words from positive sentiment rows
positive_tweet_words = [word for index, row in df.iterrows() if row['s

#create a frequency distribution of the positive words list
positive_tweet_freqdist = FreqDist(positive_tweet_words)

#list of cleaned words from negative sentiment rows
negative_tweet_words = [word for index, row in df.iterrows() if row['s

#create a frequency distribution of the negative words list
negative_tweet_freqdist = FreqDist(negative_tweet_words)

#list of cleaned words from neutral sentiment rows
neutral_tweet_words = [word for index, row in df.iterrows() if row['se

#create a frequency distribution of the neutral words list
neutral_tweet_freqdist = FreqDist(neutral_tweet_words)
```

In [48]:
```
# Create subplots for the three graphs
fig, ax = plt.subplots(1, 3, figsize=(18, 5))

# Plot the top 10 word frequency for positive sentiment
ax[0].bar(*zip(*positive_tweet_freqdist.most_common(10)))
ax[0].set_title('Positive Sentiment')

# Plot the top 10 word frequency for negative sentiment
ax[1].bar(*zip(*negative_tweet_freqdist.most_common(10)))
ax[1].set_title('Negative Sentiment')

# Plot the top 10 word frequency for neutral sentiment
ax[2].bar(*zip(*neutral_tweet_freqdist.most_common(10)))
ax[2].set_title('Neutral Sentiment')

# Set common x-label and y-label for all subplots
fig.text(0.5, 0.00, 'Word', ha='center')
fig.text(0.00, 0.5, 'Frequency', va='center', rotation='vertical')

# Adjust spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()
```

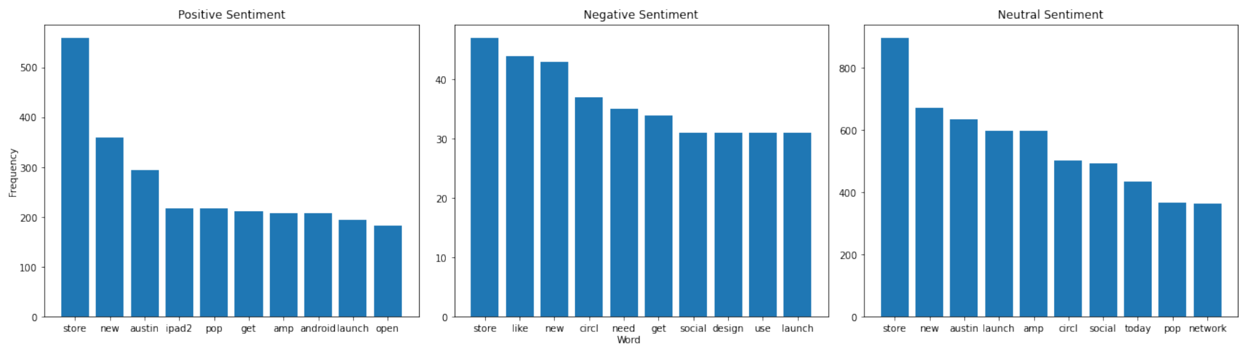


Remove Company Labels

As we can see in the graphs, there is a lot of crossover between the three different sentiments for 10 most frequent words, and most of them are of course the names of the brands and products. While the ipad2 seems to only appear in the positive sentiment as a member of the 10 most frequent, which could bet telling about that product, more inspection needs to be done. Next we removed names of the companies and broad products to see which new words would take their place in the graphs and if we would learn anything new.

In [49]:
```
#create a list of brand names
brands_list = ['ipad', 'appl', 'googl', 'iphon', 'app']

#remove them from the existing lists
brandless_positive_tweet_words = [word for word in positive_tweet_word
brandless_negative_tweet_words = [word for word in negative_tweet_word
brandless_neutral_tweet_words = [word for word in neutral_tweet_words 

#update the FreqDist action
brandless_positive_tweet_freqdist = FreqDist(brandless_positive_tweet_
brandless_negative_tweet_freqdist = FreqDist(brandless_negative_tweet_
brandless_neutral_tweet_freqdist = FreqDist(brandless_neutral_tweet_wo
```

In [50]:
```
# Create subplots for the three graphs
fig, ax = plt.subplots(1, 3, figsize=(18, 5))

# Plot the top 10 word frequency for positive sentiment
ax[0].bar(*zip(*brandless_positive_tweet_freqdist.most_common(10)))
ax[0].set_title('Positive Sentiment')

# Plot the top 10 word frequency for negative sentiment
ax[1].bar(*zip(*brandless_negative_tweet_freqdist.most_common(10)))
ax[1].set_title('Negative Sentiment')

# Plot the top 10 word frequency for neutral sentiment
ax[2].bar(*zip(*brandless_neutral_tweet_freqdist.most_common(10)))
ax[2].set_title('Neutral Sentiment')

# Set common x-label and y-label for all subplots
fig.text(0.5, 0.00, 'Word', ha='center')
fig.text(0.00, 0.5, 'Frequency', va='center', rotation='vertical')

# Adjust spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()
```



We're starting to see a little bit more information. For instance, ipad2 remains a likely indicator of a positive sentiment, while the word "need" seems to indicate the tweet is more likely to be negative. Interestingly, mention of the android also seems to more likely indicate a positive tweet.

Add Column Without Brand Names

In [51]:

```
# Define a function to remove brands from text
def remove_brands(text):
    filtered_words = [word for word in text if word not in brands_list
    return filtered_words

# Apply the remove_brands function to the 'tweet_without_stopwords' co
df['brandless_stop_tweet_stemmed'] = df['stop_tweet_stemmed'].apply(la

# Display the updated DataFrame
df.head()
```

Out[51]:

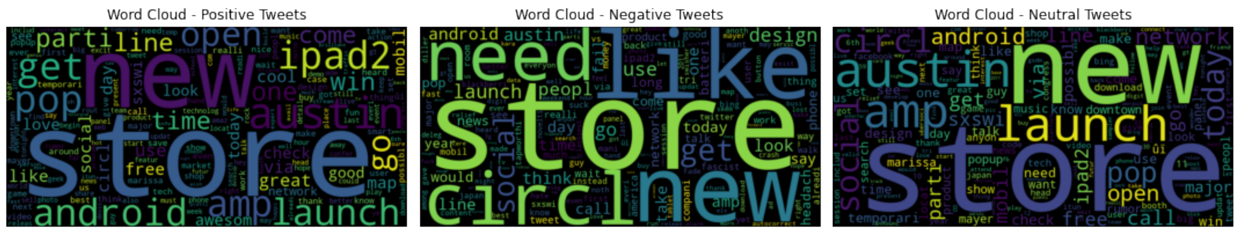
	tweet	brand_or_product	sentiment	tweet_tokenized	stop_tweet_tokenized	sto
0	.@wesley83 i have a 3g iphone. after 3 hrs tweeting at #rise_austin, it was dead! i need to upgrade. plugin stations at #sxsw.	iPhone	Negative	[wesley83, have, 3g, iphone, after, hrs, tweeting, at, rise_austin, it, was, dead, need, to, upgrade, plugin, stations, at, sxsw]	[wesley83, 3g, iphone, hrs, tweeting, rise_austin, dead, need, upgrade, plugin, stations, sxsw]	[w hr c
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	[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, fludapp, awesome, ipad, iphone, app, likely, appreciate, design, also, giving, free, ts, sxsw]	ipa ap
2	@swonderlin can not wait for #ipad 2 also. they should sale them down at #sxsw.	iPad	Positive	[swonderlin, can, not, wait, for, ipad, also, they, should, sale, them, down, at, sxsw]	[swonderlin, wait, ipad, also, sale, sxsw]	
3	@sxsw i hope this year's festival isn't as crashy as this year's iphone app.	iPad or iPhone App	Negative	[sxsw, hope, this, year, festival, isn, as, crashy, as, this, year,	[sxsw, hope, year, festival, crashy, year, iphone, app, sxsw]	

	#sxsw			iphone, app, sxsw]		
4	@sxtxstate great stuff on fri #sxsw: marissa mayer (google), tim o'reilly (tech books/conferences) &amp; matt mullenweg (wordpress)	Google	Positive	[sxtxstate, great, stuff, on, fri, sxsw, marissa, mayer, google, tim, reilly, tech, books, conferences, amp, matt, mullenweg, wordpress]	[sxtxstate, great, stuff, fri, sxsw, marissa, mayer, google, tim, reilly, tech, books, conferences, amp, matt, mullenweg, wordpress]	[sx gc

Word Cloud

We created a word cloud for each of the 3 sentiment categories, a common display of words and their frequency.

In [52]:
```
# Count the frequency of each word for each sentiment
positive_word_counts = Counter(brandless_positive_tweet_words)
negative_word_counts = Counter(brandless_negative_tweet_words)
neutral_word_counts = Counter(brandless_neutral_tweet_words)

# Generate the word cloud
positive_wordcloud = WordCloud(width=800, height=400).generate_from_fr
negative_wordcloud = WordCloud(width=800, height=400).generate_from_fr
neutral_wordcloud = WordCloud(width=800, height=400).generate_from_fre

# Create a figure and axes
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Display the word clouds with titles
axes[0].imshow(positive_wordcloud, interpolation='bilinear')
axes[0].set_title('Word Cloud - Positive Tweets')
axes[0].axis('off')

axes[1].imshow(negative_wordcloud, interpolation='bilinear')
axes[1].set_title('Word Cloud - Negative Tweets')
axes[1].axis('off')

axes[2].imshow(neutral_wordcloud, interpolation='bilinear')
axes[2].set_title('Word Cloud - Neutral Tweets')
axes[2].axis('off')

# Adjust spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()
```



We are seeing more information now. The words "new" and "store" seem to be common across the board, but "launch" is much more common in neutral tweets, "austin" is most common in positive tweets, and "like" is very common in negative tweets, which is difficult to interpret without more context, but you can also see some more indicative words like "headach", which is clearly our stemmed version of the word "headache". The presence of this word is a strong indicator that the tweet is negative.

Bigrams

Next we created bigrams to get a bit more context of the top words for each sentiment.

In [53]:
```
#create bigrams for each sentiment
positive_bigrams = list(ngrams(positive_tweet_words, 2))
negative_bigrams = list(ngrams(negative_tweet_words, 2))
neutral_bigrams = list(ngrams(neutral_tweet_words, 2))

#measure the frequency
positive_bigram_freq = Counter(positive_bigrams)
negative_bigram_freq = Counter(negative_bigrams)
neutral_bigram_freq = Counter(neutral_bigrams)

#order them in frequency
most_common_positive_bigrams = positive_bigram_freq.most_common()
most_common_negative_bigrams = negative_bigram_freq.most_common()
most_common_neutral_bigrams = neutral_bigram_freq.most_common()

#get the sum and normalize the frequency
total_positive_bigrams = sum(positive_bigram_freq.values())
total_negative_bigrams = sum(negative_bigram_freq.values())
total_neutral_bigrams = sum(neutral_bigram_freq.values())

normalized_positive_bigrams = [(bigram, freq / total_positive_bigrams
normalized_negative_bigrams = [(bigram, freq / total_negative_bigrams
normalized_neutral_bigrams = [(bigram, freq / total_neutral_bigrams *


#print first 15 rows of each
print("Top 15 Bigrams With a Positive Sentiment: ", normalized_positiv
print("Top 15 Bigrams With a Negative Sentiment: ", normalized_negativ
print("Top 15 Bigrams With a Neutral Sentiment: ", normalized_neutral_
```

```
Top 15 Bigrams With a Positive Sentiment:  [(('appl', 'store'), 0.786
9609317617432), (('iphon', 'app'), 0.5526214543038019), (('pop', 'sto
re'), 0.4826693714805358), (('appl', 'open'), 0.3917316638102899),
(('social', 'network'), 0.30079395614004406), (('googl', 'map'), 0.30
079395614004406), (('appl', 'pop'), 0.29729635199888077), (('ipad', '
app'), 0.2937987478577175), (('new', 'social'), 0.26931551886957433),
(('downtown', 'austin'), 0.2518274981637578), (('store', 'downtown'),
0.24483228988143121), (('googl', 'launch'), 0.24483228988143121), (('
temporari', 'store'), 0.23433947745794131), (('new', 'ipad'), 0.22734
426917561473), (('marissa', 'mayer'), 0.2028610401874716)]

Top 15 Bigrams With a Negative Sentiment:  [(('iphon', 'app'), 0.4423
213021939137), (('appl', 'store'), 0.4423213021939137), (('ipad', 'de
sign'), 0.3538570417551309), (('design', 'headach'), 0.30077848549186
126), (('googl', 'circl'), 0.28308563340410475), (('new', 'social'),
0.28308563340410475), (('googl', 'launch'), 0.2653927813163482), (('s
ocial', 'network'), 0.2653927813163482), (('news', 'app'), 0.24769992
922859166), (('compani', 'america'), 0.23000707714083513), (('ipad',
'news'), 0.21231422505307856), (('major', 'new'), 0.2123142250530785
6), (('fascist', 'compani'), 0.21231422505307856), (('iphon', 'batter
i'), 0.194621372965322), (('network', 'call'), 0.194621372965322)]

Top 15 Bigrams With a Neutral Sentiment:  [(('social', 'network'), 0.
6987021259654431), (('appl', 'store'), 0.680786686838124), (('new', '
social'), 0.6210685564137273), (('googl', 'launch'), 0.5593598216418
505), (('call', 'circl'), 0.49566048252249384), (('network', 'call'),
0.4876980651325743), (('major', 'new'), 0.4379329564455769), (('pop',
'store'), 0.4379329564455769), (('launch', 'major'), 0.42598933036069
75), (('appl', 'open'), 0.4259893303606975), (('possibl', 'today'),
0.3642805955882075), (('circl', 'possibl'), 0.36228999124134087),
(('googl', 'circl'), 0.31053427820686363), (('iphon', 'app'), 0.28664
70260371049), (('store', 'austin'), 0.2667409825623059)]
```

Some of the distinct pairings include tweets about downtown Austin where the convention took place in the positive sentiment camp, while tweets about ipad designs are in the negative sentiment camp.

Visualize Sentiment by Brand/Product

In this sub-section, we plotted graphs to inspect the breakdown of tweets' sentiment by each value in our brand_or_product column.

In [54]:
```
#create copy of dataframe
brand_or_product_df = df.copy()

#condense to Apple or Google
brand_or_product_df['brand'] = brand_or_product_df['brand_or_product']
                                                 'iPhone': 'Apple',
                                                 'iPad': 'Apple',
                                                 'iPad or iPhone App':
                                                 'Android': 'Google',
                                                 'Android App': 'Google
                                                 'Other Apple product o
                                                 'Other Google product 
```

In [55]:
```
#create a df that groups by brand values
brand_df = brand_or_product_df.groupby(['brand', 'sentiment'])
```

In [56]:
```
#create a df that groups by product values
product_df = brand_or_product_df.groupby(['brand_or_product', 'sentime
```

In [57]:
```
#calculate the count of each sentiment value
count_df = brand_df.size().unstack()
count_df
```

Out[57]:

sentiment	Negative	Neutral	Positive
brand			
Apple	413	2200	2085
Google	151	1747	826
Unknown	5	1428	59

In [58]:
```
#convert to percentages
percentage_df = count_df.div(count_df.sum(axis=1), axis=0) * 100
percentage_df
```

Out[58]:

sentiment	Negative	Neutral	Positive
brand			
Apple	8.790975	46.828438	44.380587
Google	5.543319	64.133627	30.323054
Unknown	0.335121	95.710456	3.954424

In [59]:
```
#calculate the count of each sentiment value
product_count_df = product_df.size().unstack()
product_count_df
```

Out[59]:

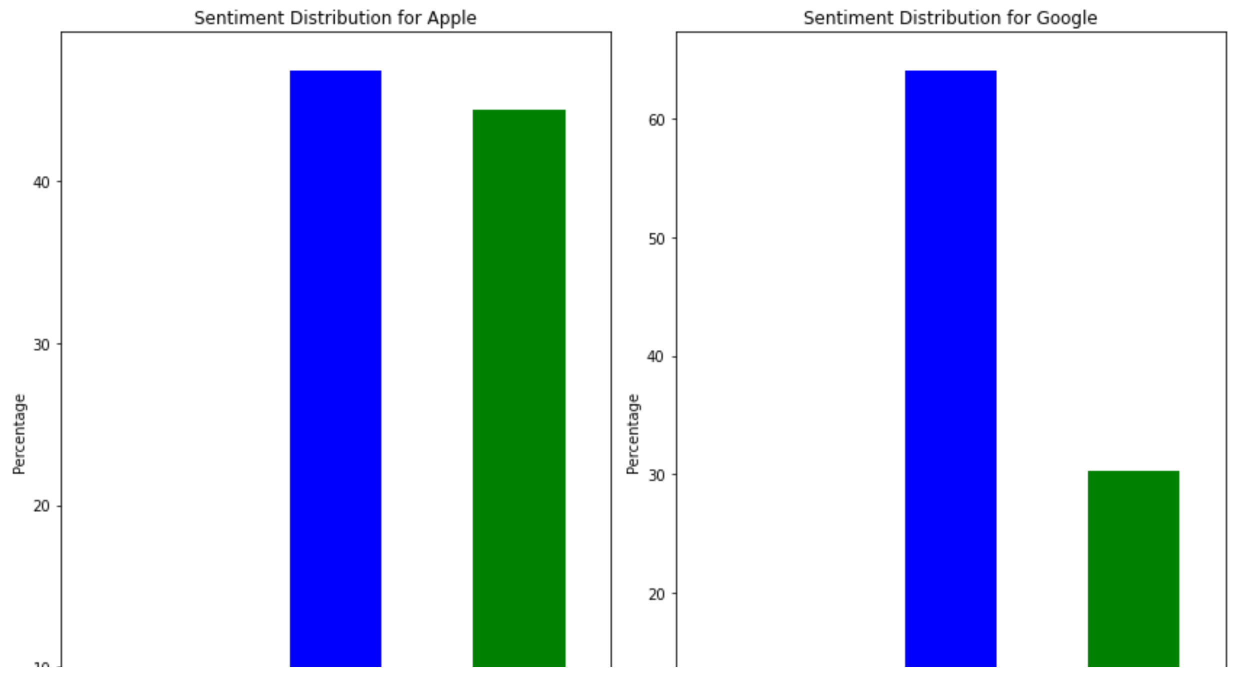
sentiment brand_or_product	Negative	Neutral	Positive
Android	10	193	83
Android App	8	1	71
Apple	99	662	567
Google	86	1544	436
Other Apple product or service	2	1	32
Other Google product or service	47	9	236
Unknown	5	1428	59
iPad	136	895	856
iPad or iPhone App	63	10	396
iPhone	113	632	234

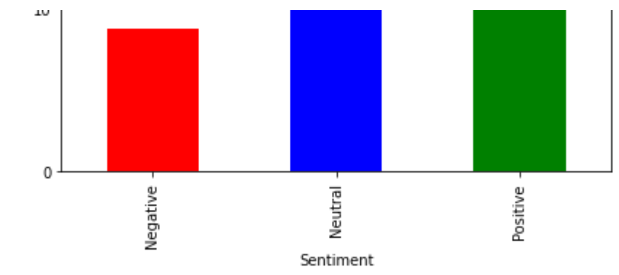
In [60]:
```
#convert to percentages
percentage_count_df = product_count_df.div(product_count_df.sum(axis=1
percentage_count_df
```

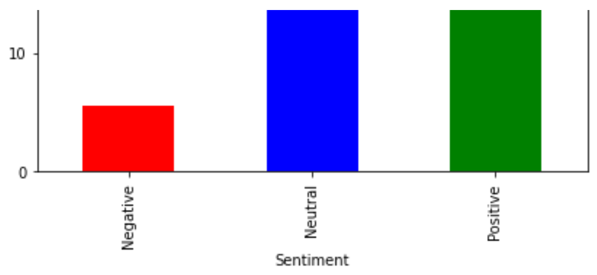
Out[60]:

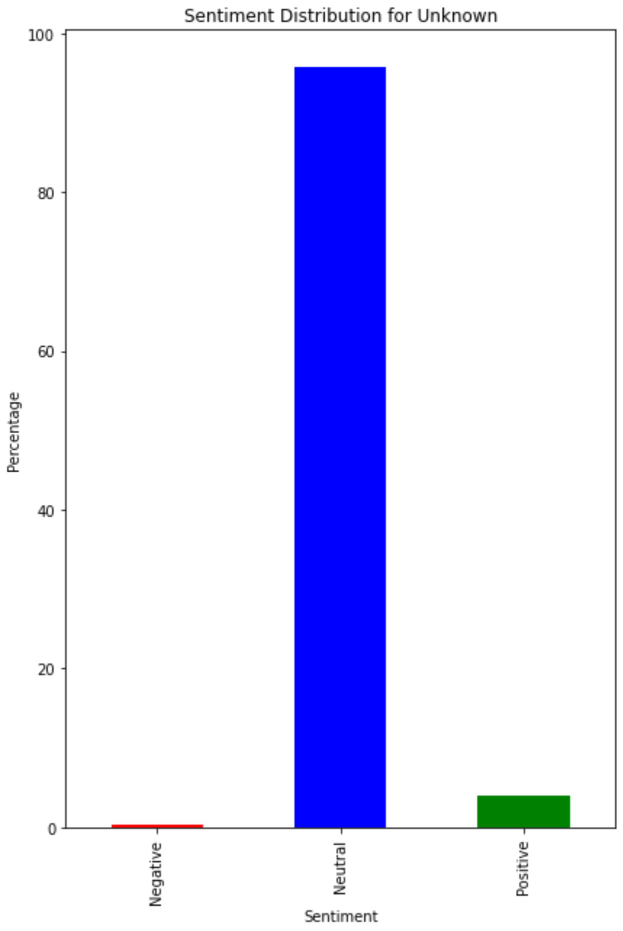
sentiment brand_or_product	Negative	Neutral	Positive
Android	3.496503	67.482517	29.020979
Android App	10.000000	1.250000	88.750000
Apple	7.454819	49.849398	42.695783
Google	4.162633	74.733785	21.103582
Other Apple product or service	5.714286	2.857143	91.428571
Other Google product or service	16.095890	3.082192	80.821918
Unknown	0.335121	95.710456	3.954424
iPad	7.207207	47.429783	45.363010
iPad or iPhone App	13.432836	2.132196	84.434968
iPhone	11.542390	64.555669	23.901941

In [61]:
```
# Calculate the number of rows and columns for the subplots
nrows = (len(percentage_df) + 1) // 2
ncols = 2

# Define colors for each sentiment value
colors = {'Positive': 'green', 'Negative': 'red', 'Neutral': 'blue'}

# Create a grid of subplots with adjusted size
fig, axes = plt.subplots(nrows, ncols, figsize=(12, 18))

# Flatten the axes array to iterate over it
axes = axes.flatten()

# Loop over each brand and plot the bar graph on a separate subplot
for i, brand in enumerate(percentage_df.index):
    percentages = percentage_df.loc[brand]
    percentages.plot.bar(ax=axes[i], color=[colors.get(sentiment, 'gra
    axes[i].set_title(f"Sentiment Distribution for {brand}")
    axes[i].set_xlabel("Sentiment")
    axes[i].set_ylabel("Percentage")

# Remove any extra subplots
if len(percentage_df) < nrows * ncols:
    for j in range(len(percentage_df), nrows * ncols):
        fig.delaxes(axes[j])

# Adjust the spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()
```









Upon inspecting these graphs, while it's less likely to come across a negative tweet than a positive/neutral tweet for all brands and products, it would appear that tweets directed at the iPhone are the most contentious. Tweets about Google in general and the Android are not overwhelmingly positive either.

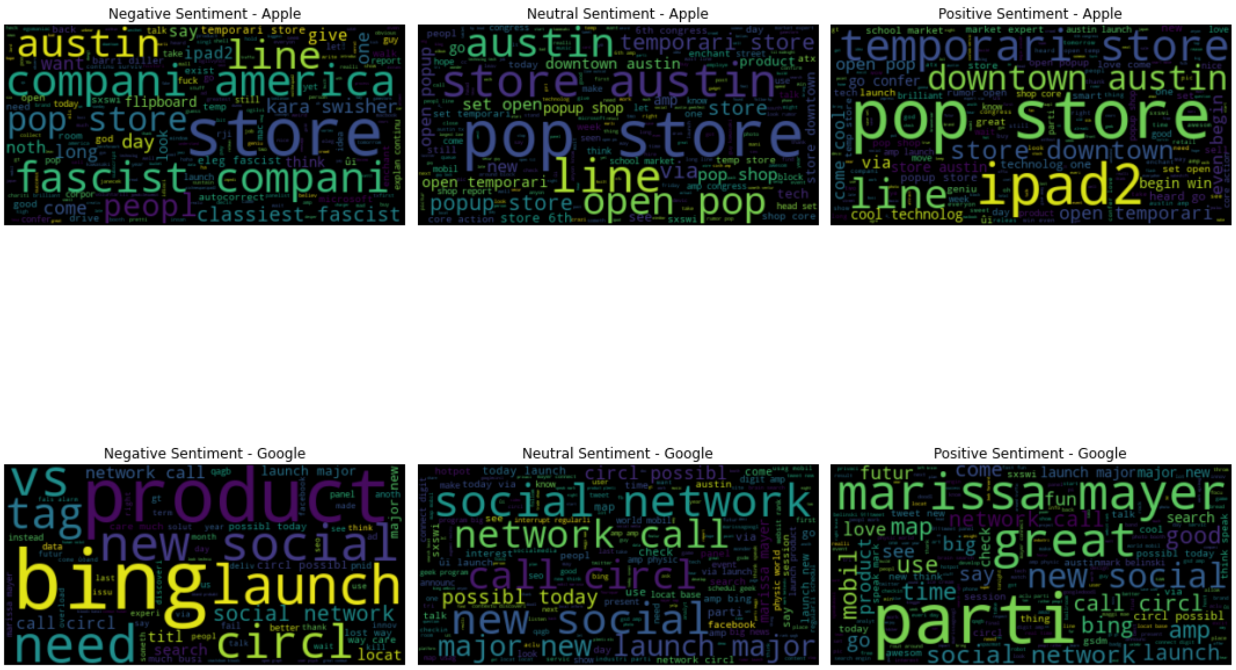
In [62]:
```
df.head()
```

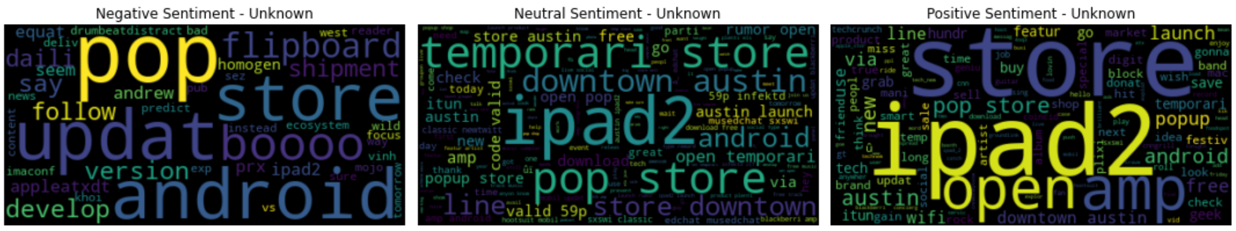
Out[62]:

	tweet	brand_or_product	sentiment	tweet_tokenized	stop_tweet_tokenized	sto
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	[wesley83, have, 3g, iphone, after, hrs, tweeting, at, rise_austin, it, was, dead, need, to, upgrade, plugin, stations, at, sxsw]	[wesley83, 3g, iphone, hrs, tweeting, rise_austin, dead, need, upgrade, plugin, stations, sxsw]	[w hr c
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	[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, fludapp, awesome, ipad, iphone, app, likely, appreciate, design, also, giving, free, ts, sxsw]	ipa ap
2	@swonderlin can not wait for #ipad 2 also. they should sale them down at #sxsw.	iPad	Positive	[swonderlin, can, not, wait, for, ipad, also, they, should, sale, them, down, at, sxsw]	[swonderlin, wait, ipad, also, sale, 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	[sxsw, hope, this, year, festival, isn, as, crashy, as, this, year, iphone, app, sxsw]	[sxsw, hope, year, festival, crashy, year, iphone, app, sxsw]	
4	@sxtxstate great stuff on fri #sxsw: marissa mayer (google), tim o'reilly (tech books/conferences) &amp; matt mullenweg (wordpress)	Google	Positive	[sxtxstate, great, stuff, on, fri, sxsw, marissa, mayer, google, tim, reilly, tech, books, conferences, amp, matt, mullenweg, wordpress]	[sxtxstate, great, stuff, fri, sxsw, marissa, mayer, google, tim, reilly, tech, books, conferences, amp, matt, mullenweg, wordpress]	[sx gc

In [63]:
```
# Calculate the number of rows and columns for the subplots
nrows = len(percentage_df)
ncols = len(percentage_df.columns)

# Create a grid of subplots with adjusted size
fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(15, 18))
```

```python
# Loop over each brand and sentiment
for i, brand in enumerate(percentage_df.index):
    for j, sentiment in enumerate(percentage_df.columns):
        # Filter the data for the specific brand and sentiment
        filtered_data = df[(df['brand_or_product'] == brand) & (df['se

        # Concatenate all the text data
        text = ' '.join(filtered_data['brandless_stop_tweet_stemmed'].

        # Generate the word cloud
        wordcloud = WordCloud().generate(text)

        # Plot the word cloud on the corresponding subplot
        axes[i, j].imshow(wordcloud, interpolation='bilinear')
        axes[i, j].set_title(f"{sentiment} Sentiment - {brand}")
        axes[i, j].axis('off')

# Remove any extra subplots
if len(percentage_df) < nrows * ncols:
    for i in range(len(percentage_df), nrows):
        for j in range(ncols):
            fig.delaxes(axes[i, j])

# Adjust the spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()
```





This provides a lot more insight into keywords that arise for each product by sentiment.

Model

This next section focuses on developing a classification model that can predict the sentiment of a tweet based on its text.

Binary Classification

First thing we do is extract the 'Positive' and 'Negative' rows to create a simple binary classification model.

In [64]:
```
#extract 'Positive' and 'Negative' rows into a new df
binary_df = df[df['sentiment'] != 'Neutral']
```

In [65]:
```
#convert 'sentiment' values to binary code
binary_df['sentiment'] = binary_df['sentiment'].replace({'Negative': 0
                                                         'Positive': 1}

#split the data into a train test split
X = binary_df['tweet']
y = binary_df['sentiment']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

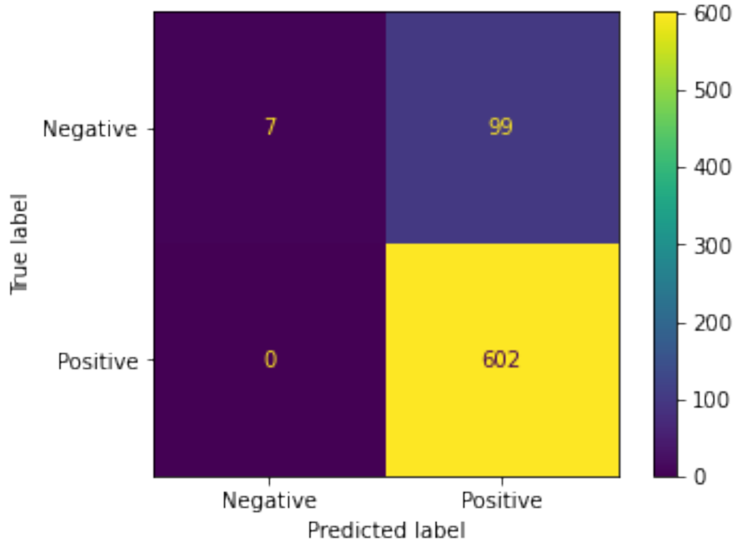
In [66]:
```
#instantiate tokenizer function
tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True)
```

In [67]:
```
# Create the Pipeline Naive Bayes model
clf_pipe = Pipeline([
        ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                       stop_words=stopwords_list)),
        ('clf', MultinomialNB(alpha=1.0))])

#Fit Model
clf_pipe.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf_pipe.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
```

```
Accuracy: 0.8601694915254238
Precision: 0.8587731811697575
Recall: 1.0
F1-Score: 0.9240214888718342
```

The classification report scores seem pretty good! Next we plotted a confusion matrix of our model's results.

In [68]:
```
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix
plot_confusion_matrix(clf_pipe, X_test, y_test, display_labels=['Negat
plt.show()
```



In [69]:
```
#check the 'sentiment' values for imbalance inspection
binary_df['sentiment'].value_counts(normalize=True)
```

Out[69]:
```
1    0.83922
0    0.16078
Name: sentiment, dtype: float64
```

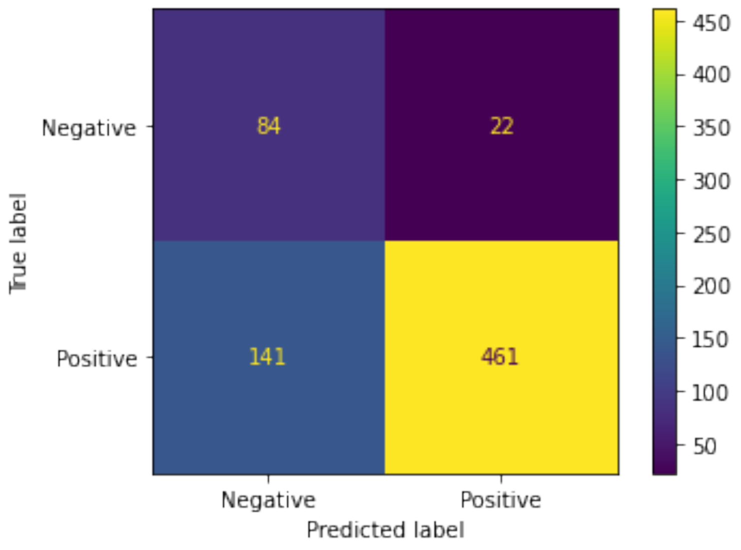
Upon further review, the model is predicting nearly every tweet as having a positive sentiment. Due to a class imbalance that overwhelmingly favors the 'Positive' value for the 'sentiment' column, our metrics are misleading. We need to address the class imbalance.

In [70]:
```python
# Create the Pipeline Naive Bayes model with undersampler added
clf_pipe = Pipeline([
        ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                       stop_words=stopwords_list)),
        ('undersampler', RandomUnderSampler(random_state=42)),
        ('clf', MultinomialNB(alpha=1.0))])

#Fit Model
clf_pipe.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf_pipe.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
```

```
Accuracy: 0.769774011299435
Precision: 0.9544513457556936
Recall: 0.7657807308970099
F1-Score: 0.8497695852534562
```

Accuracy and Recall have dropped, but Precision has risen and F1 only experience a slight dip. Let's see how our confusion matrix looks now, as well as a classification report.

In [71]:
```
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix
plot_confusion_matrix(clf_pipe, X_test, y_test, display_labels=['Negat
plt.show()
```



In [72]:
```
#create classification report
report = classification_report(y_test, y_pred)

#print classification report
print(report)
```

```
              precision    recall  f1-score   support

           0       0.37      0.79      0.51       106
           1       0.95      0.77      0.85       602

    accuracy                           0.77       708
   macro avg       0.66      0.78      0.68       708
weighted avg       0.87      0.77      0.80       708
```

It still has a tough time with negative sentiment, but these are much better results!

HyperParameter Tuning

Next we tuned our model with GridSearchCV.

In [73]:
```
#define parameter grid
param_grid = {
    'clf__alpha': [0.01, 0.05, 0.1, 0.5, 1],
    'clf__fit_prior': [True, False]
}

#Create a GridSearchCV object
grid_search = GridSearchCV(estimator=clf_pipe, param_grid=param_grid,

#Fit the GridSearchCV object to data
grid_search.fit(X_train, y_train)

#Print the best hyperparameters and best score
print("Best Hyperparameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)

#Print the best model trained on the entire dataset
best_model = grid_search.best_estimator_
print("Best Model: ", best_model)
```

```
Best Hyperparameters:  {'clf__alpha': 1, 'clf__fit_prior': True}
Best Score:  0.7414474545216594
Best Model:  Pipeline(steps=[('vectorizer',
                 TfidfVectorizer(stop_words=['i', 'me', 'my', 'mysel
f', 'we',
                                             'our', 'ours', 'ourselve
s', 'you',
                                             "you're", "you've", "yo
u'll",
                                             "you'd", 'your', 'your
s',
                                             'yourself', 'yourselve
s', 'he',
                                             'him', 'his', 'himself',
'she',
                                             "she's", 'her', 'hers',
'herself',
                                             'it', "it's", 'its', 'it
self', ...],
                                 tokenizer=<bound method TweetTokeniz
er.tokenize of <nltk.tokenize.casual.TweetTokenizer object at 0x7face
f139f70>>)),
                ('undersampler', RandomUnderSampler(random_state=4
2)),
                ('clf', MultinomialNB(alpha=1))])
```

Our gridsearch resulted in default parameters, so no need for hypertuning. Let's try a more complex model for comparison.

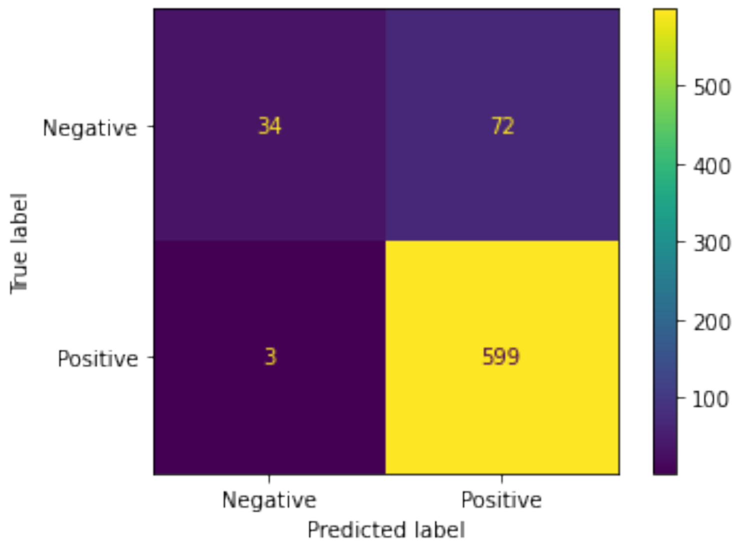
Random Forest

Our next model for comparison is a Random Forest model.

In [74]:
```
#create random forest pipeline model
forest_pipe = Pipeline([
        ('vectorizer', TfidfVectorizer(stop_words=stopwords_list,
                                       tokenizer=tokenizer.tokenize)),
        ('clf', RandomForestClassifier(class_weight='balanced', random

#fit model onto data
forest_pipe.fit(X_train, y_train)

#predict results
y_pred = forest_pipe.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
```

```
Accuracy: 0.8940677966101694
Precision: 0.8926974664679582
Recall: 0.9950166112956811
F1-Score: 0.9410840534171248
```

In [75]:
```
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix
plot_confusion_matrix(forest_pipe, X_test, y_test, display_labels=['Ne
plt.show()
```



In [76]:
```
#create classification report
report = classification_report(y_test, y_pred)

#print classification report
print(report)
```

```
              precision    recall  f1-score   support

           0       0.92      0.32      0.48       106
           1       0.89      1.00      0.94       602

    accuracy                           0.89       708
   macro avg       0.91      0.66      0.71       708
weighted avg       0.90      0.89      0.87       708
```

Similar to our Naive Bayes model, our baseline Random Forest model is doing a good job of predicting positive sentiment, but struggling with negative predictions. Now we'll proceed to getting the best tuning, as well as add in our undersampler.

In [77]:
```
#define parameter grid
forest_param_grid = {
    'clf__n_estimators': [100, 200, 300],
    'clf__criterion': ['gini', 'entropy'],
    'clf__max_depth': [None, 5, 10]
}

#Create a GridSearchCV object
grid_search = GridSearchCV(estimator=forest_pipe, param_grid=forest_pa

#Fit the GridSearchCV object to data
grid_search.fit(X_train, y_train)

#Print the best hyperparameters and best score
print("Best Hyperparameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)

#Print the best model trained on the entire dataset
best_model = grid_search.best_estimator_
print("Best Model: ", best_model)
```

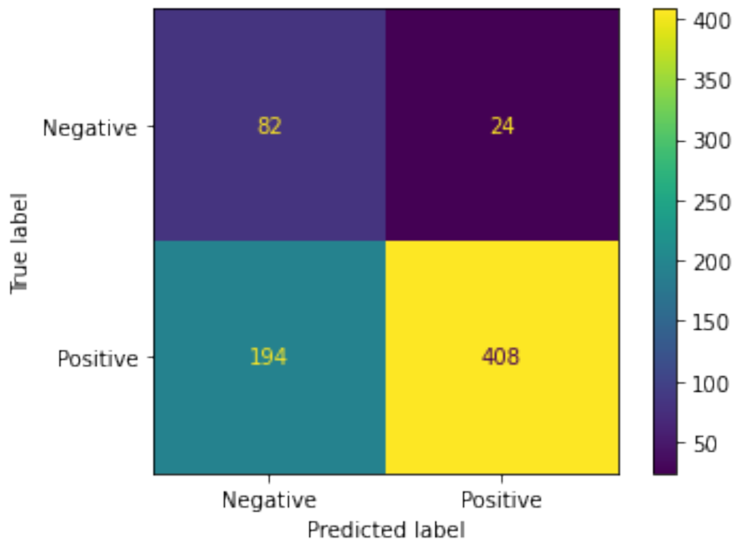
```
Best Hyperparameters:  {'clf__criterion': 'gini', 'clf__max_depth': N
one, 'clf__n_estimators': 300}
Best Score:  0.8675366599983796
Best Model:  Pipeline(steps=[('vectorizer',
                 TfidfVectorizer(stop_words=['i', 'me', 'my', 'mysel
f', 'we',
                                             'our', 'ours', 'ourselve
s', 'you',
                                             "you're", "you've", "yo
u'll",
                                             "you'd", 'your', 'your
s',
                                             'yourself', 'yourselve
s', 'he',
                                             'him', 'his', 'himself',
'she',
                                             "she's", 'her', 'hers',
'herself',
                                             'it', "it's", 'its', 'it
self', ...],
                                 tokenizer=<bound method TweetTokeniz
er.tokenize of <nltk.tokenize.casual.TweetTokenizer object at 0x7facf
0ea8fd0>>)),
                ('clf',
                 RandomForestClassifier(class_weight='balanced',
                                        n_estimators=300, random_stat
e=42))])
```

Hypertuning the random forest model results in default settings, so there's no need for tuning the model but we'll add in our sampler.

In [78]:
```
#create random forest pipeline model
forest_pipe = Pipeline([
        ('vectorizer', TfidfVectorizer(stop_words=stopwords_list,
                                       tokenizer=tokenizer.tokenize)),
        ('undersampler', RandomUnderSampler(random_state=42)),
        ('clf', RandomForestClassifier(class_weight='balanced', random

#fit model onto data
forest_pipe.fit(X_train, y_train)

#predict results
y_pred = forest_pipe.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
```

```
Accuracy: 0.692090395480226
Precision: 0.9444444444444444
Recall: 0.6777408637873754
F1-Score: 0.7891682785299807
```

In [79]:
```python
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix
plot_confusion_matrix(forest_pipe, X_test, y_test, display_labels=['Ne
plt.show()
```



In [80]:
```python
#create classification report
report = classification_report(y_test, y_pred)

#print classification report
print(report)
```

```
              precision    recall  f1-score   support

           0       0.30      0.77      0.43       106
           1       0.94      0.68      0.79       602

    accuracy                           0.69       708
   macro avg       0.62      0.73      0.61       708
weighted avg       0.85      0.69      0.74       708
```

Final Model

After comparing the two models, both baseline and tuned, it appears that our hypertuned Naive Bayes model performs the strongest for our task of binary classification.

- Higher correct number of negative and positive predictions.
- Higher macro precision, recall, and f1 scores, as well as accuracy.
- Higher precision and recall scores for both negative and positive predictions.

Multiclass Classification

Our final section of machine learning will deal with building a model that can not only predict 'Positive' and 'Negative' sentiment, but 'Neutral' as well.

In [81]:
```
#create a copy of the original dataframe
model_df = df.copy()

#check our three values are present
print(model_df['sentiment'].unique())
```

```
['Negative' 'Positive' 'Neutral']
```

In [82]:
```
#numerize sentiment values
model_df['sentiment'] = model_df['sentiment'].replace({
                                                        'Negative': 0,
                                                        'Positive': 1,
                                                        'Neutral': 2})

#check for imbalance
print(model_df['sentiment'].value_counts(normalize=True))
```

```
2    0.602984
1    0.333184
0    0.063832
Name: sentiment, dtype: float64
```

Now we're already aware that we will have an imbalance problem with our modeling that we'll need to address.

In [83]:
```
#train test split
X = model_df['tweet']
y = model_df['sentiment']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state
```

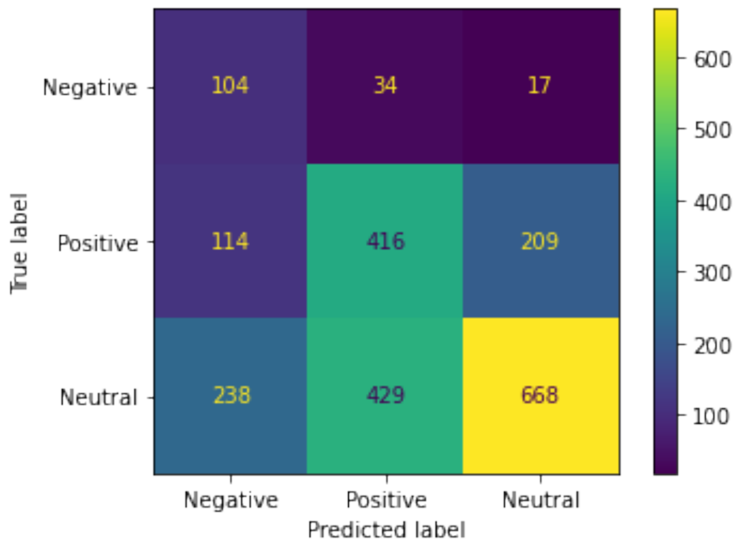
Multinomial Model

In [84]:
```
# Create the Pipeline Naive Bayes model with undersampler added
multi_clf_pipe = imblearn.pipeline.Pipeline([
        ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                       stop_words=stopwords_list)),
        ('undersampler', RandomUnderSampler(random_state=42)),
        ('clf', MultinomialNB(alpha=1.0))])

#Fit Model
multi_clf_pipe.fit(X_train, y_train)

# Make predictions on the test set
y_pred = multi_clf_pipe.predict(X_test)
```

In [85]:
```
#create classification report
report = classification_report(y_test, y_pred)

#print classification report
print(report)
```

```
              precision    recall  f1-score   support

           0       0.23      0.67      0.34       155
           1       0.47      0.56      0.51       739
           2       0.75      0.50      0.60      1335

    accuracy                           0.53      2229
   macro avg       0.48      0.58      0.48      2229
weighted avg       0.62      0.53      0.55      2229
```

In [86]:
```
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix
plot_confusion_matrix(multi_clf_pipe, X_test, y_test, display_labels=[
plt.show()
```



In [87]:
```
#define parameter grid
param_grid = {
    'clf__alpha': [0.01, 0.05, 0.1, 0.5, 1],
    'clf__fit_prior': [True, False]
}

#Create a GridSearchCV object
grid_search = GridSearchCV(estimator=multi_clf_pipe, param_grid=param_

#Fit the GridSearchCV object to data
grid_search.fit(X_train, y_train)

#Print the best hyperparameters and best score
print("Best Hyperparameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)

#Print the best model trained on the entire dataset
best_model = grid_search.best_estimator_
print("Best Model: ", best_model)
```

```
Best Hyperparameters:  {'clf__alpha': 1, 'clf__fit_prior': True}
Best Score:  0.4978309648466716
Best Model:  Pipeline(steps=[('vectorizer',
                    TfidfVectorizer(stop_words=['i', 'me', 'my', 'mysel
f', 'we',
                                                 'our', 'ours', 'ourselve
s', 'you',
                                                 "you're", "you've", "yo
u'll",
                                                 "you'd", 'your', 'your
s',
                                                 'yourself', 'yourselve
s', 'he',
                                                 'him', 'his', 'himself',
'she',
                                                 "she's", 'her', 'hers',
'herself',
                                                 'it', "it's", 'its', 'it
self', ...],
                                 tokenizer=<bound method TweetTokeniz
er.tokenize of <nltk.tokenize.casual.TweetTokenizer object at 0x7face
e6808b0>>)),
                ('undersampler', RandomUnderSampler(random_state=4
2)),
                ('clf', MultinomialNB(alpha=1))])
```

It appears our MultinomialNB does pretty well across the board, and our gridsearch results in default parameters, so there is no need for hypertuning. The model's biggest struggle is differentiating between positive and neutral sentiment in tweets.

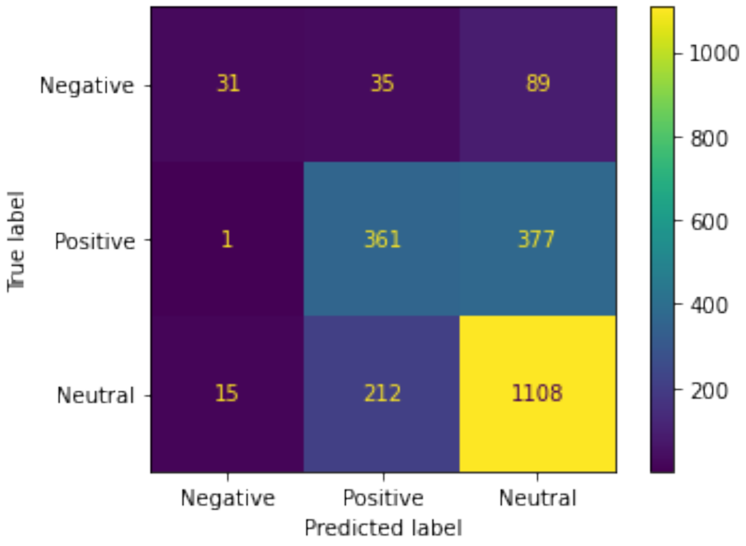
Multiclass Random Forest

In [88]:
```
#create random forest pipeline model
multi_forest_pipe = Pipeline([
        ('vectorizer', TfidfVectorizer(stop_words=stopwords_list,
                                       tokenizer=tokenizer.tokenize)),
        ('clf', RandomForestClassifier(class_weight='balanced', random

#fit model onto data
multi_forest_pipe.fit(X_train, y_train)

#predict results
y_pred = multi_forest_pipe.predict(X_test)
```

In [89]:
```
#create classification report
report = classification_report(y_test, y_pred)

#print classification report
print(report)
```

```
              precision    recall  f1-score   support

           0       0.66      0.20      0.31       155
           1       0.59      0.49      0.54       739
           2       0.70      0.83      0.76      1335

    accuracy                           0.67      2229
   macro avg       0.65      0.51      0.53      2229
weighted avg       0.66      0.67      0.66      2229
```

In [90]:
```python
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix
plot_confusion_matrix(multi_forest_pipe, X_test, y_test, display_label
plt.show()
```



Our Random Forest baseline performs much worse than our Naive Bayes. Let's tune it.

In [91]:
```
#define parameter grid
forest_param_grid = {
    'clf__n_estimators': [100, 200, 300],
    'clf__criterion': ['gini', 'entropy'],
    'clf__max_depth': [None, 5, 10]
}

#Create a GridSearchCV object
grid_search = GridSearchCV(estimator=multi_forest_pipe, param_grid=for

#Fit the GridSearchCV object to data
grid_search.fit(X_train, y_train)

#Print the best hyperparameters and best score
print("Best Hyperparameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)

#Print the best model trained on the entire dataset
best_model = grid_search.best_estimator_
print("Best Model: ", best_model)
```

```
Best Hyperparameters:  {'clf__criterion': 'entropy', 'clf__max_dept
h': None, 'clf__n_estimators': 300}
Best Score:  0.6768885564697082
Best Model:  Pipeline(steps=[('vectorizer',
                 TfidfVectorizer(stop_words=['i', 'me', 'my', 'mysel
f', 'we',
                                             'our', 'ours', 'ourselve
s', 'you',
                                             "you're", "you've", "yo
u'll",
                                             "you'd", 'your', 'your
s',
                                             'yourself', 'yourselve
s', 'he',
                                             'him', 'his', 'himself',
'she',
                                             "she's", 'her', 'hers',
'herself',
                                             'it', "it's", 'its', 'it
self', ...],
                                 tokenizer=<bound method TweetTokeniz
er.tokenize of <nltk.tokenize.casual.TweetTokenizer object at 0x7facf
0eba370>>)),
                ('clf',
                 RandomForestClassifier(class_weight='balanced',
                                        criterion='entropy', n_estima
tors=300,
                                        random_state=42))])
```

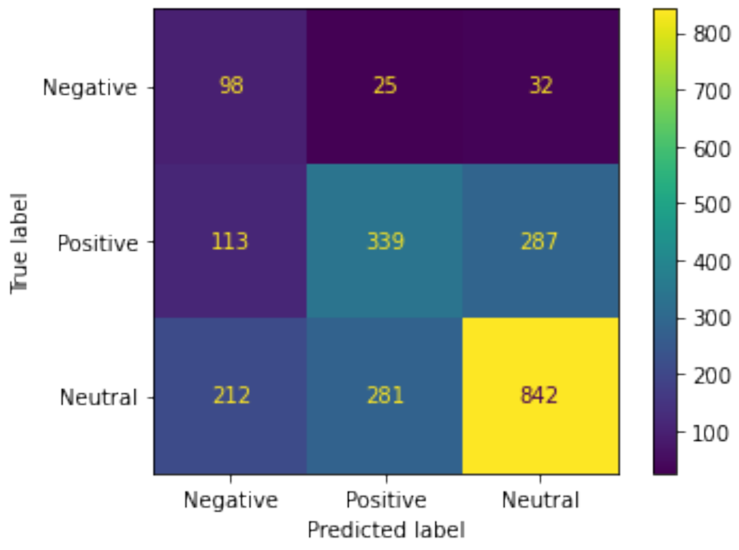
Now we run the tuned model and add in the undersampler.

In [92]:
```
#create tuned random forest pipeline model
tuned_forest_pipe = Pipeline([
        ('vectorizer', TfidfVectorizer(stop_words=stopwords_list,
                                       tokenizer=tokenizer.tokenize)),
        ('undersampler', RandomUnderSampler(random_state=42)),
        ('clf', RandomForestClassifier(class_weight='balanced', criter

#fit model onto data
tuned_forest_pipe.fit(X_train, y_train)

#predict results
y_pred = tuned_forest_pipe.predict(X_test)
```

In [93]:
```
#create classification report
report = classification_report(y_test, y_pred)

#print classification report
print(report)
```

```
              precision    recall  f1-score   support

           0       0.23      0.63      0.34       155
           1       0.53      0.46      0.49       739
           2       0.73      0.63      0.67      1335

    accuracy                           0.57      2229
   macro avg       0.49      0.57      0.50      2229
weighted avg       0.62      0.57      0.59      2229
```

In [94]:
```
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix
plot_confusion_matrix(tuned_forest_pipe, X_test, y_test, display_label
plt.show()
```



Final Model

It appears that once again, our Naive Bayes model does the best performance for multiclass prediction as well.

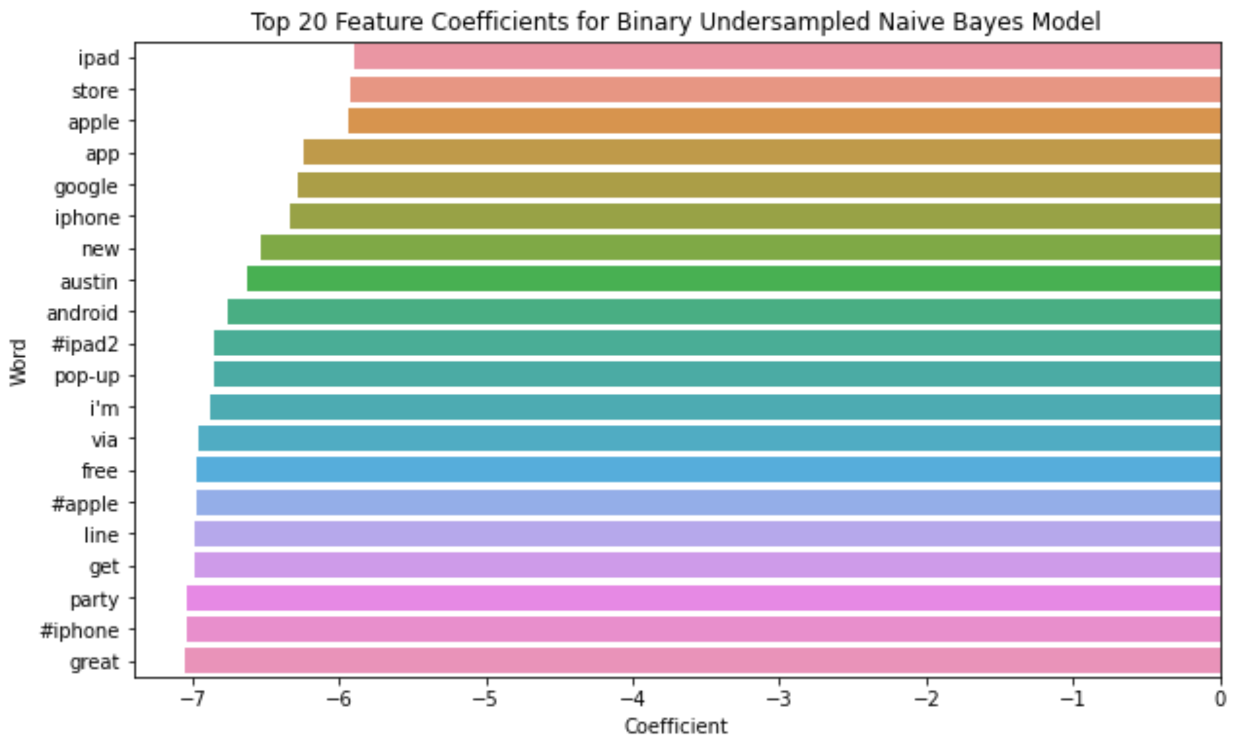
- Higher number of correct predictions of positive and negative sentiment.
- Higher macro recall score.
- Significantly higher recall score on negative predictions, while all scores are in line or higher than our Random Forest model.

Interpretation

Our next section deals with interpreting our results.

In [95]:
```
#create function to plot feature importance
def plot_importance(clf_pipe, n_features, title):
    # Extract feature names
    vectorizer = clf_pipe['vectorizer']
    feature_names = vectorizer.get_feature_names()

    # Get the MultinomialNB classifier from the pipeline
    clf = clf_pipe['clf']

    # Get the coefficients
    coefs = clf.coef_[0]

    #Get the Importance
    importance = math.e**(abs(coefs))

    # Create a dataframe
    feature_df = pd.DataFrame({'Word': feature_names, 'Coefficient': c

    # Sort feature importance
    feature_importance = feature_df.sort_values(by='Coefficient', asce

    # Plot the feature importance
    plt.figure(figsize=(10, 6))
    sns.barplot(x='Coefficient', y='Word', data=feature_importance)
    plt.xlabel('Coefficient')
    plt.ylabel('Word')
    plt.title(f'Top {n_features} Feature Coefficients for {title}')
    plt.show()
```

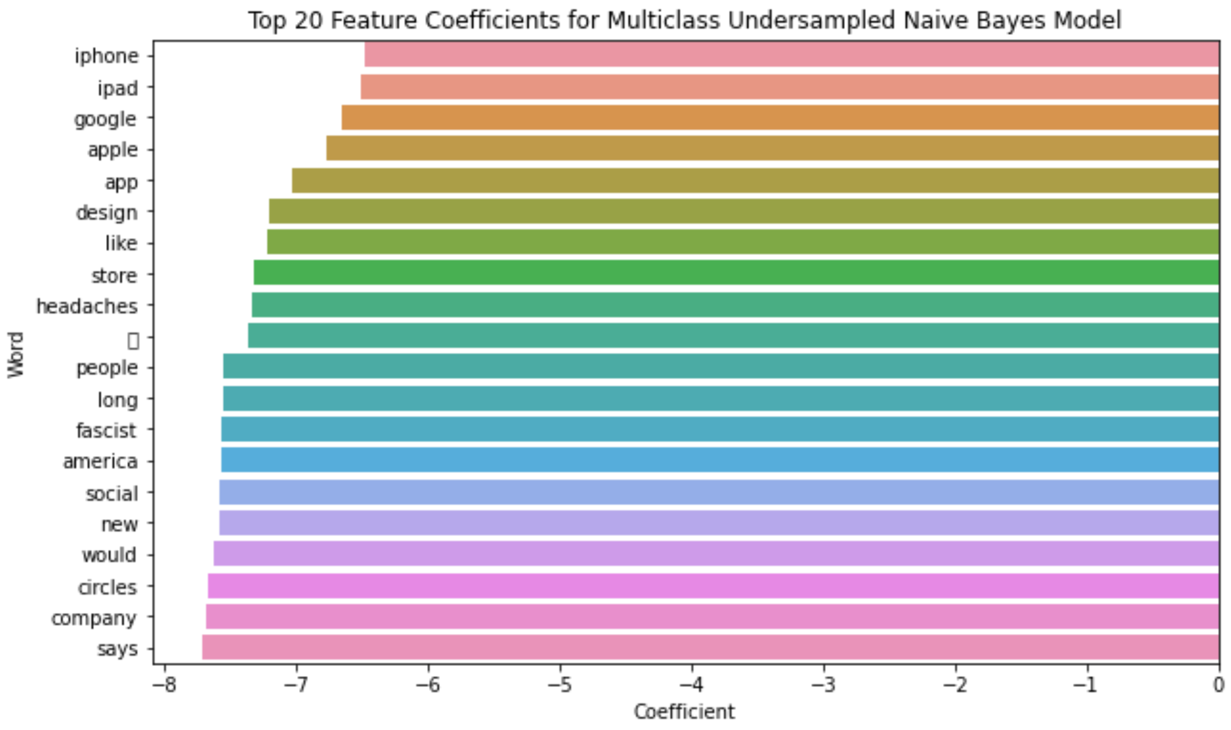
```
In [96]: plot_importance(clf_pipe, 20, 'Binary Undersampled Naive Bayes Model')
```



As one might expect, certain words like "great" and "party" pushed the model towards a positive prediction. As witnessed earlier, the ipad2 continues to have a positive sentiment attached to it, as well as mentions of the pop-up store.

In [97]:
```
plot_importance(multi_clf_pipe, 20, 'Multiclass Undersampled Naive Bay
```

```
/Users/Chris/anaconda3/envs/learn-env/lib/python3.8/site-packages/mat
plotlib/backends/backend_agg.py:238: RuntimeWarning: Glyph 137 missin
g from current font.
  font.set_text(s, 0.0, flags=flags)
/Users/Chris/anaconda3/envs/learn-env/lib/python3.8/site-packages/mat
plotlib/backends/backend_agg.py:201: RuntimeWarning: Glyph 137 missin
g from current font.
  font.set_text(s, 0, flags=flags)
```



In our multiclass model, the more impactful words include "fascist", "headaches", and "circles", words that the company should be on the lookout for in tweets as they may suggest person behind those tweets is more passionately opinionated.

Conclusions & Recommendations

Conclusions

1. Apple vs Google

- Apple has more negative tweets than Google (8.8% to 5.5%), but also has significantly more positive tweets (44% to 30%).
- Apps are very popular for both companies: 89% positive tweets about Android apps, 84% positive tweets about iPhone/iPad apps
- The iPhone has more negative tweets than the Android: 12% to 3%
- General services related to Apple are significantly more popular than Google: 91% to 80%

2. Apple Positive vs Negative Tweets

- There was a lot of enthusiasm for the pop-up store in downtown Austin.
- There was a lot of enthusiasm about the launch of the iPad2, that seems to have been very successful.
- The battery of the iPhone and the design of the iPad are frequently mentioned in negative tweets.
- The term "fascist" is used in reference to Apple in some obviously negative tweets.

3. Google Positive vs Negative Tweets

- Marissa Mayer is referenced in a positive light in tweets.
- Google's "Circle" is mentioned frequently, there's seemingly a lot of energy surrounding this from each direction.
- Some users are critical of the cost of Google products, such as Google TV.

Recommendations

1. Apple

- Improve upon the design of the iPad.
- The pop-up store was very successful, look into more strategies similar to this.

2. Google

- Marissa Mayer is a respected individual in majority of social media activity, she should be regarded as such.
- Look at ways to cut down on the prices of higher end tech products.

Next Steps

- Gather more data. We only had about 9,000 tweets in total, with a large chunk of them being marked neutral and making for a smaller sample size for our binary classifier.
- Harness more negative tweets. The imbalance in which there are thousands more neutral tweets than negative tweets really don't help us with making assessments.
- Continue to tweak the models and explore different types of models not used here.