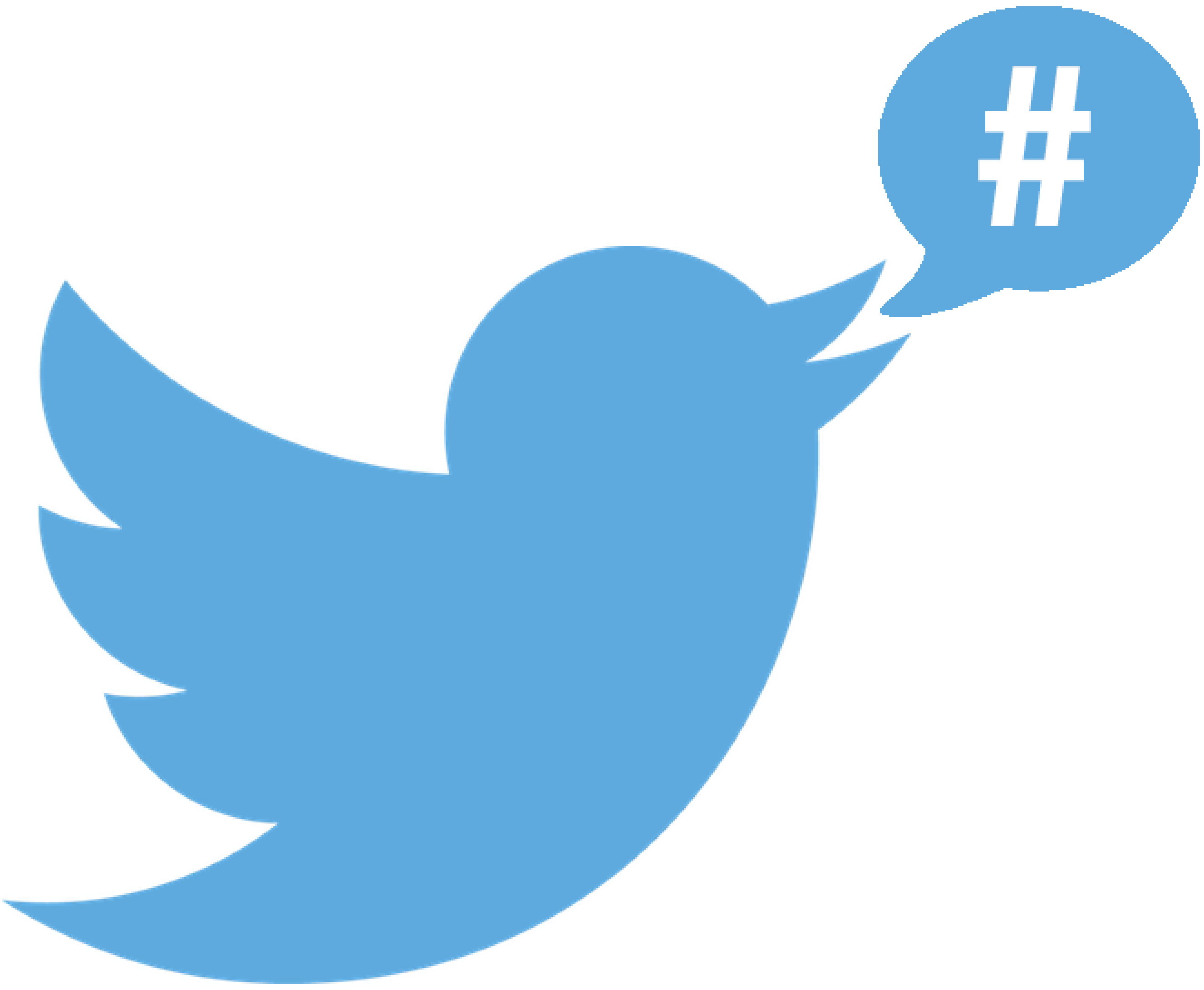
Twitter Sentiment Analysis

Authors: Aaron Onserio, Diana Mwaura, Joshua Rwanda, Samuel Kyalo, Stella Kitur, Stephanie Mbithe

TMs: Lucille Kaleha, Nikita Njoroge, Samuel Karu

Date submitted: 22/06/2023 [Thursday]

Github Repo :[here (https://github.com/R3TR0Quan/twitter-sentiment-analysis)](https://github.com/R3TR0Quan/twitter-sentiment-analysis)



Business Understanding

In today's technology-driven era, businesses understand the importance of comprehending customer perceptions and adapting to market changes. Social media platforms like Twitter provide a valuable tool for tracking and analyzing user sentiments regarding different products.

Problem Statement

In the era of social media, it is crucial for businesses to understand the sentiments expressed by customers towards their brands or products. This sentiment analysis project aims to analyze Twitter data and extract valuable insights regarding the sentiments associated with Apple and Google products mentioned in tweets. By uncovering the public's opinions and emotions, businesses can make data-driven decisions, that is relevant to the customers needs, this will then improve their market positioning and enhance customer satisfaction

Objectives

Our main objective is to create a model that when given a tweet or series of tweets and a product would determine how the user felt about that product.

This is trivial for a human to accomplish, our model can do this for thousands or even millions of tweets in a short time.

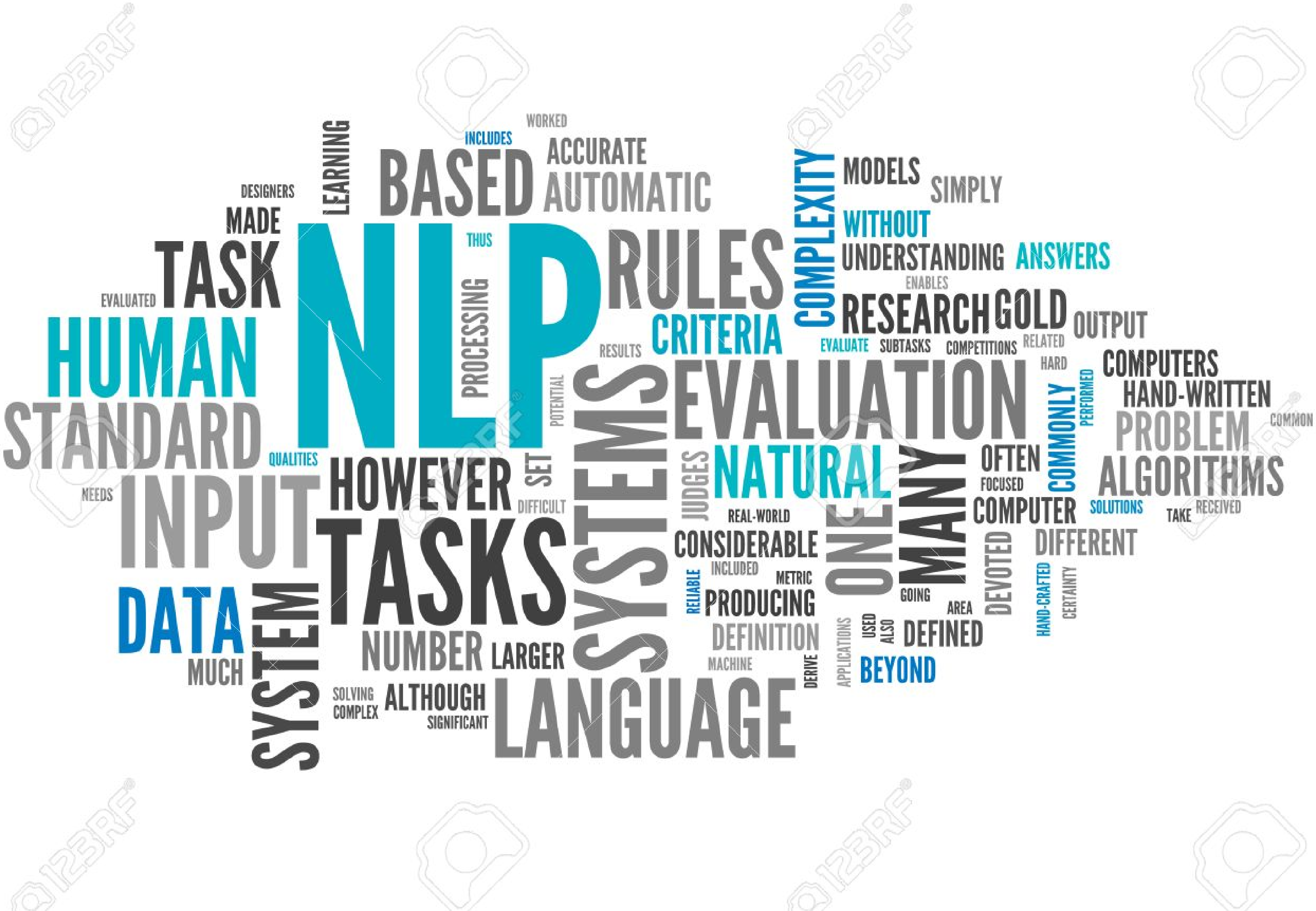
1. To build a text classifier to accurately distinguish between positive, neutral, and negative sentiments.
2. Competitive Analysis – Compare the sentiment towards Apple and Google products to identify any significant differences in public perception.
3. Give insights as to where the company can increase customer satisfaction.

Success Metrics

The model performance of this project will be analyzed using the following performance metrics:

1. Accuracy
2. Precision and Recall
3. F1 Score

For the multiclass classification, the recall macro average was used to compare the models. The recall macro average is useful in scenarios where you want to evaluate the overall performance of the classifier across all classes, regardless of the class distribution. It provides an aggregated measure of how well the model is able to identify positive instances for each class. A weakness of this metric is that it fails to take into account the number of true negatives or false positives.



1. Importing Libraries.

In [1]:
```
#Import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import string
import re
import nltk
import itertools
import imblearn.pipeline
import urllib
import requests
import warnings
warnings.filterwarnings('ignore')

from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk import TweetTokenizer
from nltk import FreqDist
from nltk.collocations import BigramAssocMeasures, BigramCollocationFinder

from imblearn.over_sampling import RandomOverSampler
from wordcloud import WordCloud,ImageColorGenerator
from PIL import Image
from xgboost import XGBClassifier

from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import recall_score, precision_score, accuracy_score

from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.dummy import DummyClassifier
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegressionCV
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
```

2. Loading Data.

In [2]:
```
# Loading the data
data = pd.read_csv('Data/crowdflower_tweet_data.csv', encoding = 'latin1')
data.head()
```

Out[2]:

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_or_product
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

3. Data Understanding

The dataset used in this project is from data.world provided by CrowdFlower which has tweets about Apple and Google from the South by Southwest (SXSW) conference. The tweet labels were crowdsourced and reflect which emotion they convey and what product/service/company this emotion is directed at based on the content. The dataset is made up of 9093 rows and 3 columns. Our target column is 'is_there_an_emotion_directed_at_a_brand_or_product'. There are 22 duplicated values in the dataset and missing values as well. All the columns are of the object data type.

In [3]:
```
# Checking the shape to see how many columns and rows our data has
data.shape
```

Out[3]:
```
(9093, 3)
```

In [4]:
```
# Cheching on the names of the columns
data.columns
```

Out[4]:
```
Index(['tweet_text', 'emotion_in_tweet_is_directed_at',
       'is_there_an_emotion_directed_at_a_brand_or_product'],
      dtype='object')
```

In [5]:
```python
# Mapping the items in the column 'emotion_in_tweet_is_directed_at' in ascending order
data['emotion_in_tweet_is_directed_at'].value_counts(normalize=True)
```

Out[5]:
```
iPad                               0.287451
Apple                              0.200851
iPad or iPhone App                 0.142814
Google                             0.130659
iPhone                             0.090246
Other Google product or service    0.089031
Android App                        0.024613
Android                            0.023701
Other Apple product or service     0.010635
Name: emotion_in_tweet_is_directed_at, dtype: float64
```

In [6]:
```python
# Mapping the content in the column 'is_there_an_emotion_directed_at_a_brand_or_product'
data['is_there_an_emotion_directed_at_a_brand_or_product'].value_counts(normalize=True)
```

Out[6]:
```
No emotion toward brand or product    0.592654
Positive emotion                      0.327505
Negative emotion                      0.062686
I can't tell                          0.017156
Name: is_there_an_emotion_directed_at_a_brand_or_product, dtype: float64
```

In [7]:
```python
# Checking the general information in the dataset
data.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 [8]:
```python
# Checking for the missing values
data.isna().sum()
```

Out[8]:
```
tweet_text                                               1
emotion_in_tweet_is_directed_at                       5802
is_there_an_emotion_directed_at_a_brand_or_product       0
dtype: int64
```

We can see that the "emotion_in_tweet_is_directed_at" column has the most number of missing values.

In [9]:
```python
# Check for the duplicated entries
data.duplicated().sum()
```

Out[9]:
```
22
```

In [10]:
```python
# Veiw the duplicated entries in the DataFrame
data[data.duplicated()].head()
```

Out[10]:

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_or_product
468	Before It Even Begins, Apple Wins #SXSW {link}	Apple	Positive emotion
776	Google to Launch Major New Social Network Call...	NaN	No emotion toward brand or product
2232	Marissa Mayer: Google Will Connect the Digital...	NaN	No emotion toward brand or product
2559	Counting down the days to #sxsw plus strong Ca...	Apple	Positive emotion
3950	Really enjoying the changes in Gowalla 3.0 for...	Android App	Positive emotion

4. Data Cleaning

This section prepares the data for EDA and modeling. The missing values in the dataset are dealt with by replacing the NaN values by 'Unknown Product'.

Duplicate values are dropped from the dataset and we retained the first entry.

The columns also have long names that do not make any sense, so we will go ahead and rename them and give them titles that makes more sense and more undestandable.

In [11]:
```
# Renaming columns
data.columns = ['Tweet', 'Product/Brand', 'Emotion']
data
```

Out[11]:

	Tweet	Product/Brand	Emotion
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
...	...	...	...
9088	Ipad everywhere. #SXSW {link}	iPad	Positive emotion
9089	Wave, buzz... RT @mention We interrupt your re...	NaN	No emotion toward brand or product
9090	Google's Zeiger, a physician never reported po...	NaN	No emotion toward brand or product
9091	Some Verizon iPhone customers complained their...	NaN	No emotion toward brand or product
9092	Ï¡ Ïà ü_ Ê Î Ò £ Á ââ _ £ â_ ÛâRT @...	NaN	No emotion toward brand or product

9093 rows × 3 columns

Since there is only one missing value in the column `Tweet` we will drop it. The missing values in the column `Product/Bran` ' will be replaced with **'Unknown'** for better mapping of the the product entries. For the duplicated entries, we will drop the duplications and **keep first** entrie only.

In [12]:
```
# Dropping the missing value in the subset 'Tweet'
data = data.dropna(subset=['Tweet'])
```

In [13]:
```
# Filling the missing values with "Unknown"
data['Product/Brand'].fillna("Unknown Product", inplace = True)
data.isna().sum()
```

Out[13]:
```
Tweet            0
Product/Brand    0
Emotion          0
dtype: int64
```

In [14]:
```
# Dropping the duplicated values and keeping first entries
data.drop_duplicates(keep='first', inplace=True)
```

In the column of `Emotion` we will replace the sentiments with shorter and more understandable names

In [15]:
```
emotion = {'Positive emotion': 'Positive', 'Negative emotion': 'Negative',
                'No emotion toward brand or product': 'Neutral',
                "I can't tell": 'Unknown'}
data['Emotion'] = data['Emotion'].map(emotion)
data.head()
```

Out[15]:

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

In [16]:
```
# Counting the values on the column `Emotion`
data['Emotion'].value_counts()
```

Out[16]:
```
Neutral     5375
Positive    2970
Negative     569
Unknown      156
Name: Emotion, dtype: int64
```

In [17]:
```
pd.set_option("display.max_colwidth", 300)
data[data['Emotion']=='Unknown']
```

Out[17]:

	Tweet	Product/Brand	Emotion
90	Thanks to @mention for publishing the news of @mention new medical Apps at the #sxswi conf. blog {link} #sxsw #sxswh	Unknown Product	Unknown
102	ÛÏ@mention "Apple has opened a pop-up store in Austin so the nerds in town for #SXSW can get their new iPads. {link} #wow	Unknown Product	Unknown
237	Just what America needs. RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw	Unknown Product	Unknown
341	The queue at the Apple Store in Austin is FOUR blocks long. Crazy stuff! #sxsw	Unknown Product	Unknown
368	Hope it's better than wave RT @mention Buzz is: Google's previewing a social networking platform at #SXSW: {link}	Unknown Product	Unknown
...	...	...	...
9020	It's funny watching a room full of people hold their iPad in the air to take a photo. Like a room full of tablets staring you down. #SXSW	Unknown Product	Unknown
9032	@mention yeah, we have @mention , Google has nothing on us :) #SXSW	Unknown Product	Unknown
9037	@mention Yes, the Google presentation was not exactly what I was expecting. #sxsw	Unknown Product	Unknown
9058	"Do you know what Apple is really good at? Making you feel bad about your Xmas present!" - Seth Meyers on iPad2 #sxsw #doyoureallyneedthat?	Unknown Product	Unknown
9066	How much you want to bet Apple is disproportionately stocking the #SXSW pop-up store with iPad 2? The influencer/hipsters thank you	Apple	Unknown

156 rows × 3 columns

In [18]:
```
# Dropping the "Uknown" sentiment
data = data[data['Emotion']!='Unknown']
data['Emotion'].value_counts()
```

Out[18]:
```
Neutral     5375
Positive    2970
Negative     569
Name: Emotion, dtype: int64
```

For pre-processing the tokenization is done using the TweetTokenizer, stopwords are removed using the default list of stopwords. Stemming is performed using the PorterStemmer.

In [19]:
```
# Create an instance of Regexptoenizer
tokenizer = TweetTokenizer("(?u)\w{3,}",
                          strip_handles=True)


#Create a list of stop words
stopwords_list = stopwords.words('english')

#Create an instance of the PorterStemmer
stemmer =PorterStemmer()
```

In [20]:
```
# Creating a function that will help in cleaning the dataset
def preprocess_text(text, tokenizer, stopwords_list, stemmer):
    text = text.lower()
    token = tokenizer.tokenize(text)
    stop_words = [word for word in token if word not in stopwords_list]
    punctuation_removed = [word for word in stop_words if word not in string.punctuation]
    stemmed_stopwords = [stemmer.stem(word) for word in punctuation_removed]
    return stemmed_stopwords
```

In [21]:
```
text_data = data.Tweet.apply(lambda x: preprocess_text(x, tokenizer,stopwords_list,stemmer))
text_data

data["preprocessed_text"] = text_data
data
```

Out[21]:

	Tweet	Product/Brand	Emotion	preprocessed_text
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	[3g, iphon, 3, hr, tweet, #rise_austin, dead, need, upgrad, plugin, station, #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	[know, awesom, ipad, iphon, app, like, appreci, design, also, they'r, give, free, ts, #sxsw]
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	iPad	Positive	[wait, #ipad, 2, 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	[hope, year', festiv, crashi, year', iphon, app, #sxsw]
4	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	Google	Positive	[great, stuff, fri, #sxsw, marissa, mayer, googl, tim, o'reilli, tech, book, confer, matt, mullenweg, wordpress]
...	...	...	...	...
9088	Ipad everywhere. #SXSW {link}	iPad	Positive	[ipad, everywher, #sxsw, link]
9089	Wave, buzz... RT @mention We interrupt your regularly scheduled #sxsw geek programming with big news {link} #google #circles	Unknown Product	Neutral	[wave, buzz, ..., rt, interrupt, regularli, schedul, #sxsw, geek, program, big, news, link, #googl, #circl]
9090	Google's Zeiger, a physician never reported potential AE. Yet FDA relies on physicians. "We're operating w/out data." #sxsw #health2dev	Unknown Product	Neutral	[google', zeiger, physician, never, report, potenti, ae, yet, fda, reli, physician, we'r, oper, w, data, #sxsw, #health2dev]
9091	Some Verizon iPhone customers complained their time fell back an hour this weekend. Of course they were the New Yorkers who attended #SXSW.	Unknown Product	Neutral	[verizon, iphon, custom, complain, time, fell, back, hour, weekend, cours, new, yorker, attend, #sxsw]
9092	Ï¡ Ïà ü_ Ê Î Ò £ Á ââ _ £ â_ ÛâRT @mention Google Tests ÛÏCheck-in Offers Û At #SXSW {link}	Unknown Product	Neutral	[, ï, ¡, , ïà, , ü_, , , ê, , , î, , , ò, , , £, , , á, , ââ, , , , , £, , , , , â_, , ûârt, googl, test, , ûïcheck-in, offer, , û, , #sxsw, link]

8914 rows × 4 columns

5. Exploratory Data Analysis

This section aims at retrieving more insights for analysis. The positive, negative and neutral values in the dataset are plotted on a histogram.

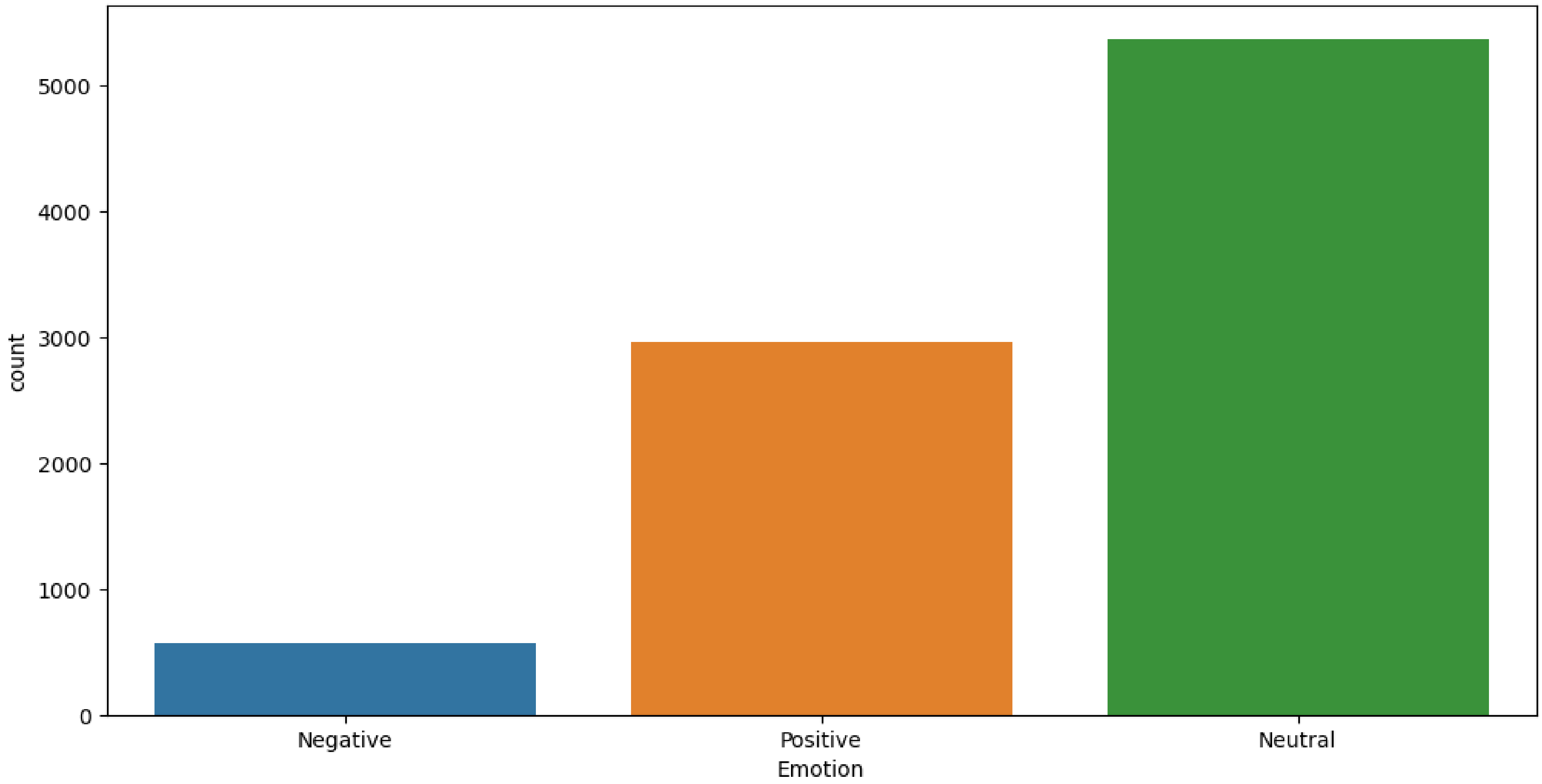
Frequency distributions of the top 10 tokens of the various emotions and the entire dataset were visualized.

In [22]:
```
# Sentiments -plot
pos_sentiments = data[data['Emotion']=='Positive']
#verifying that neutral and negative tweets have been removed
pos_sentiments['Emotion'].value_counts()
```

Out[22]:
```
Positive    2970
Name: Emotion, dtype: int64
```

In [23]:
```
# Plot a figure to visualize the quantities in the Emotion column
plt.figure(figsize=(12,6))
sns.countplot(x='Emotion',data=data)
```

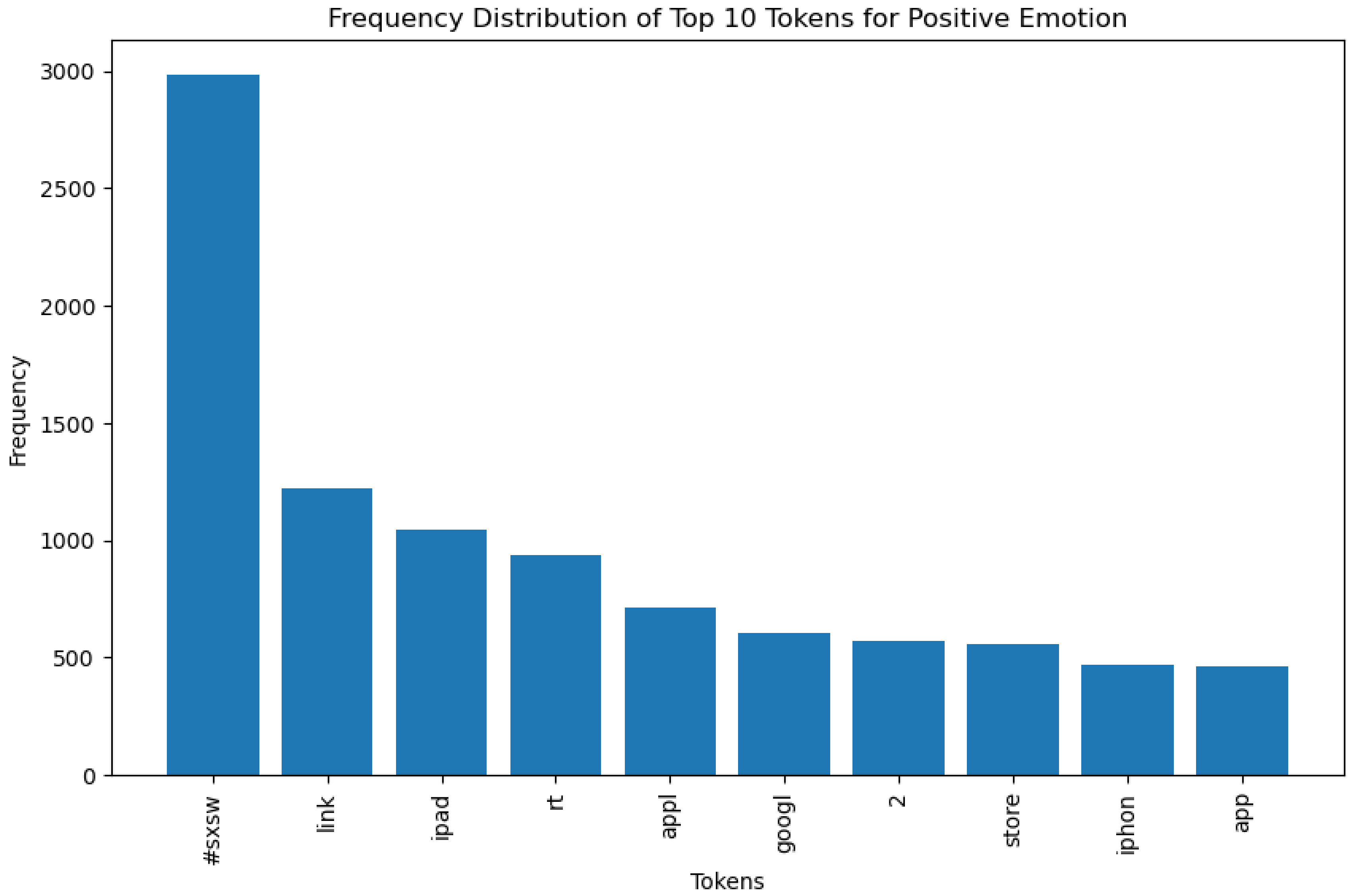
Out[23]: <AxesSubplot:xlabel='Emotion', ylabel='count'>



As shown from the above visualization, Neutral values are the highest followed by Positive values then Negative values.

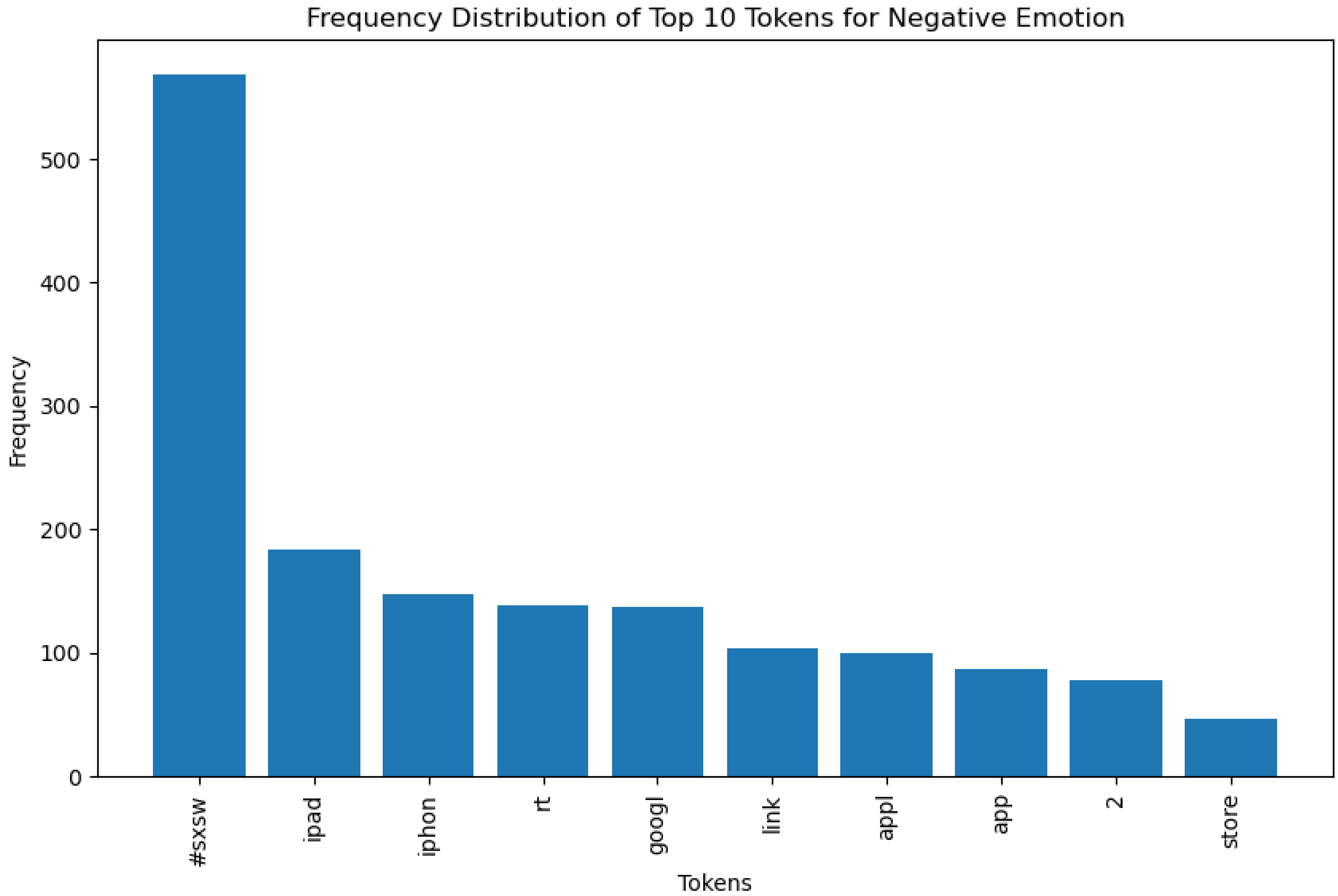
In [24]:
```
# Filter the dataframe for positive, negative, and neutral emotions
positive_tweets = data[data['Emotion'] == 'Positive']['preprocessed_text']
negative_tweets = data[data['Emotion'] == 'Negative']['preprocessed_text']
neutral_tweets = data[data['Emotion'] == 'Neutral']['preprocessed_text']

# Concatenate all positive, negative, and neutral tweets into separate lists
all_positive_tokens = [token for sublist in positive_tweets for token in sublist]
all_negative_tokens = [token for sublist in negative_tweets for token in sublist]
all_neutral_tokens = [token for sublist in neutral_tweets for token in sublist]

# Create frequency distributions for positive, negative, and neutral tokens
positive_freq_dist = FreqDist(all_positive_tokens)
negative_freq_dist = FreqDist(all_negative_tokens)
neutral_freq_dist = FreqDist(all_neutral_tokens)

# Get the top 10 most common positive, negative, and neutral tokens and their frequencies
top_positive_tokens = positive_freq_dist.most_common(10)
top_negative_tokens = negative_freq_dist.most_common(10)
top_neutral_tokens = neutral_freq_dist.most_common(10)

tokens_positive, frequencies_positive = zip(*top_positive_tokens)
tokens_negative, frequencies_negative = zip(*top_negative_tokens)
tokens_neutral, frequencies_neutral = zip(*top_neutral_tokens)
```

In [25]:
```
# Plot the frequency distribution of the most common tokens for positive emotion
plt.figure(figsize=(10, 6))
plt.bar(tokens_positive, frequencies_positive)
plt.xlabel('Tokens')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of Top 10 Tokens for Positive Emotion')
plt.xticks(rotation=90)
plt.show()
```



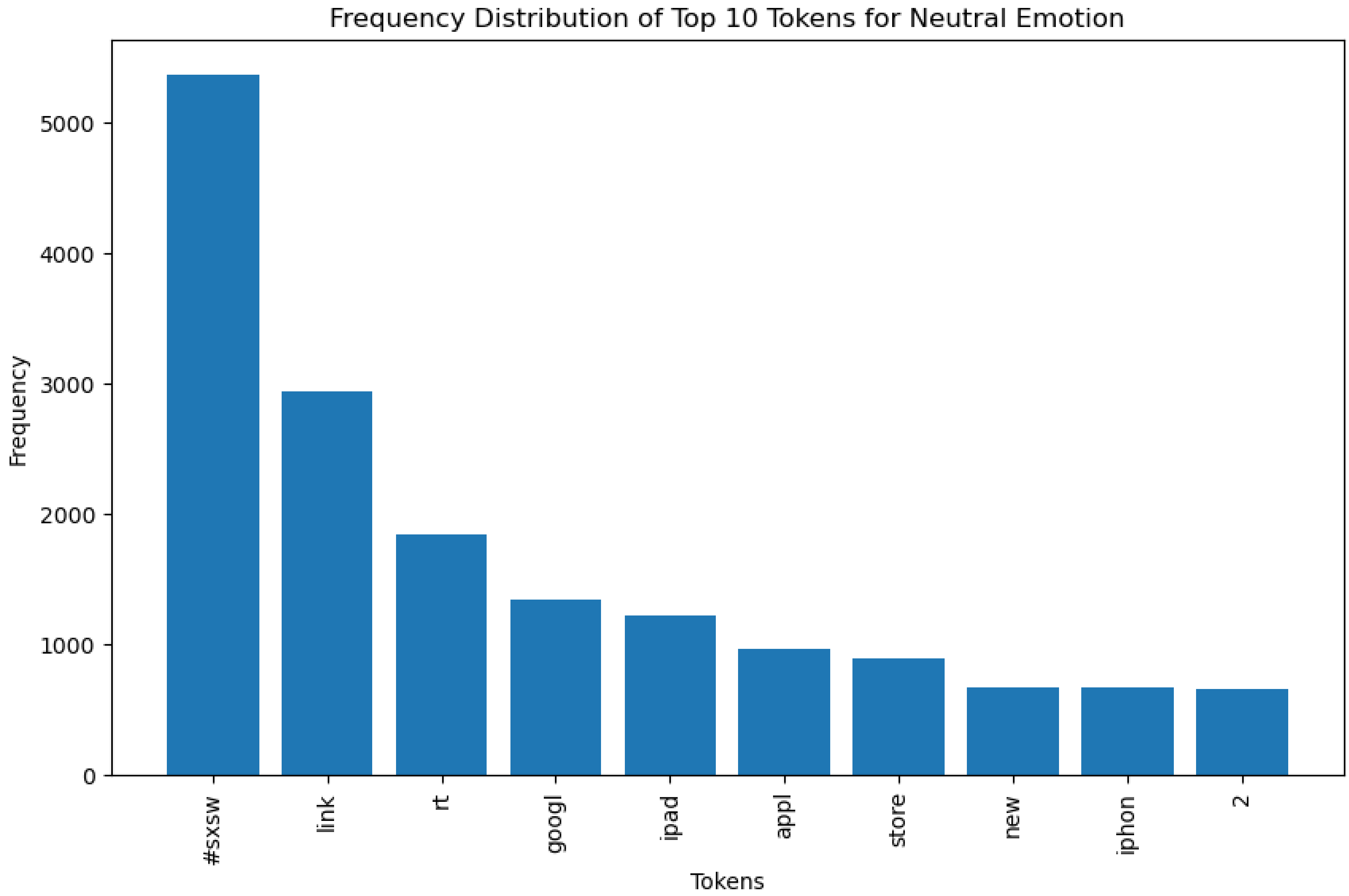
The most common words in the tweets classified as positive emotions are #sxsw (South by South West), link, ipad.

In [26]:
```
# Plot the frequency distribution of the most common tokens for negative emotion
plt.figure(figsize=(10, 6))
plt.bar(tokens_negative, frequencies_negative)
plt.xlabel('Tokens')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of Top 10 Tokens for Negative Emotion')
plt.xticks(rotation=90)
plt.show()
```



The most common words in the tweets classified as negative emotions are #sxsw (South by South West),ipad, iphone.

In [27]:
```
# Plot the frequency distribution of the most common tokens for neutral emotion
plt.figure(figsize=(10, 6))
plt.bar(tokens_neutral, frequencies_neutral)
plt.xlabel('Tokens')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of Top 10 Tokens for Neutral Emotion')
plt.xticks(rotation=90)
plt.show()
```



The most common words in the tweets classified as neutral emotions are #sxsw (South by South West), link, rt.

In [28]:
```
# Concatenate all preprocessed_text into a single list
all_tokens = [token for sublist in data["preprocessed_text"] for token in sublist]

# Create a frequency distribution of tokens
freq_dist = FreqDist(all_tokens)

# Get the top 10 most common tokens and their frequencies
top_tokens = freq_dist.most_common(10)
tokens, frequencies = zip(*top_tokens)
# Plot the frequency distribution in a bar graph
plt.figure(figsize=(10, 6))
plt.bar(tokens, frequencies)
plt.xlabel('Tokens')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of Top 10 Tokens in the entire dataset')
plt.xticks(rotation=90)
plt.show()
```

The most common words in all the tweets are #sxsw (South by South West), link, rt.

Feature Engineering

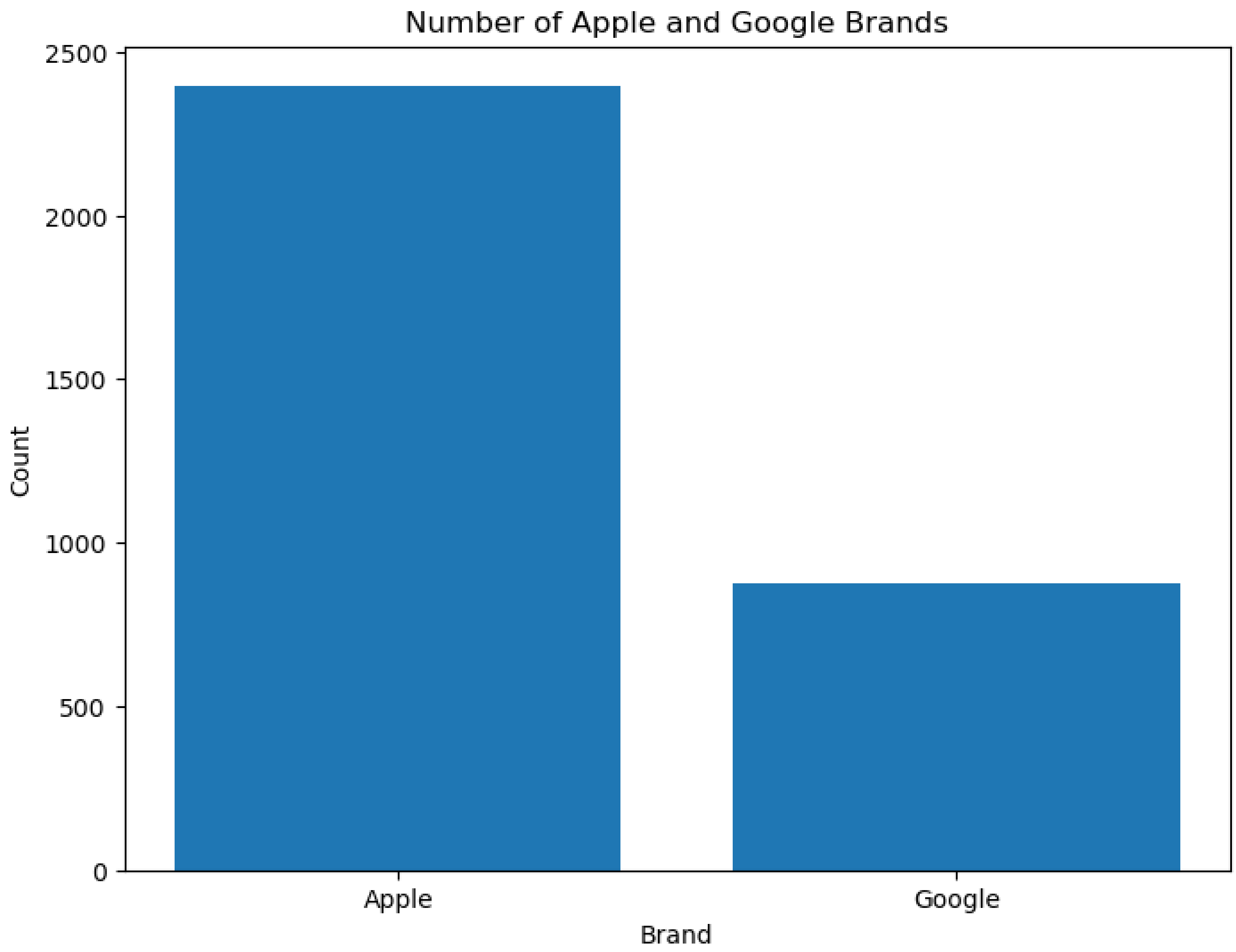
Grouping the Apple and Google products together.

In [29]:
```
#mapping products to brands
brand_dict={'iPhone': 'Apple', 'iPad or iPhone App': 'Apple', 'iPad': 'Apple',
            'Google': 'Google', 'Unknown': 'Unknown',
            'Android': 'Google', 'Apple': 'Apple', 'Android App': 'Google',
            'Other Google product or service': 'Google',
            'Other Apple product or service': 'Apple'}
data['Brand'] = data['Product/Brand'].map(brand_dict)
data['Brand'].unique()
data.head()
```

Out[29]:

	Tweet	Product/Brand	Emotion	preprocessed_text	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	Negative	[3g, iphon, 3, hr, tweet, #rise_austin, dead, need, upgrad, plugin, station, #sxsw]	Apple
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	[know, awesom, ipad, iphon, app, like, appreci, design, also, they'r, give, free, ts, #sxsw]	Apple
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	iPad	Positive	[wait, #ipad, 2, also, sale, #sxsw]	Apple
3	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	iPad or iPhone App	Negative	[hope, year', festiv, crashi, year', iphon, app, #sxsw]	Apple
4	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	Google	Positive	[great, stuff, fri, #sxsw, marissa, mayer, googl, tim, o'reilli, tech, book, confer, matt, mullenweg, wordpress]	Google

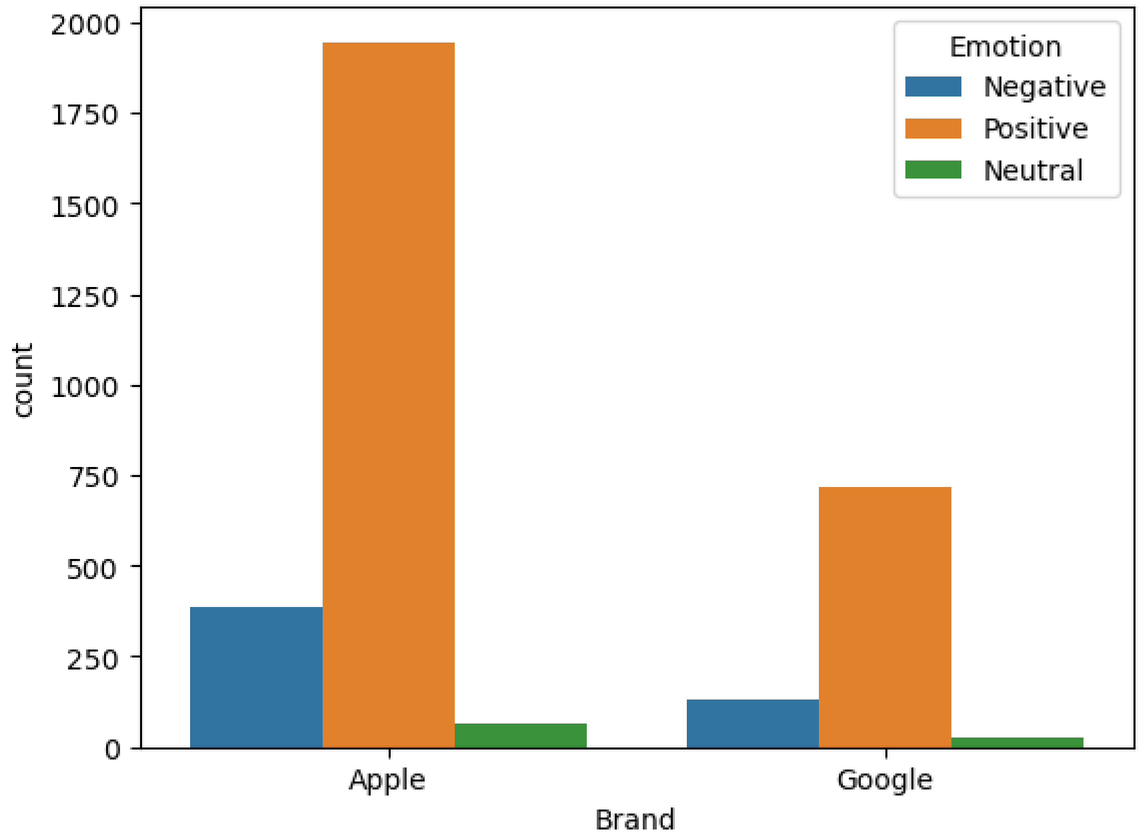
In [30]:
```
# Counts the number of Apple and Google brands
apple_count = data[data['Brand'] == 'Apple'].shape[0]
google_count = data[data['Brand'] == 'Google'].shape[0]

# Creates a bar graph of the brand counts
brands = ['Apple', 'Google']
counts = [apple_count, google_count]

plt.figure(figsize=(8, 6))
plt.bar(brands, counts)
plt.xlabel('Brand')
plt.ylabel('Count')
plt.title('Number of Apple and Google Brands')
plt.show()
```



The Apple products are more compared to the Google brand.

In [31]:
```
sns.countplot(data=data, x="Brand", hue="Emotion")
```

Out[31]: <AxesSubplot:xlabel='Brand', ylabel='count'>



For the Apple brand the Positive emotions were highest followed by the negative emotion then the neutral emotion. For the Google brand the positive emotions were highest followed by the negative emotion then the neutral being the least.

In [32]:
```
bigram_measures = BigramAssocMeasures()

# Flatten the list of lists into a single list
word_list = list(itertools.chain.from_iterable(data['preprocessed_text']))

tweet_finder = BigramCollocationFinder.from_words(word_list)
tweet_scored = tweet_finder.score_ngrams(bigram_measures.raw_freq)
tweet_scored[:50]
```

Out[32]:
```
[(('ipad', '2'), 0.010138056191625415),
 (('link', '#sxsw'), 0.008104744078216005),
 (('#sxsw', 'link'), 0.007629671154522219),
 (('#sxsw', 'rt'), 0.006175948008019231),
 (('appl', 'store'), 0.005140289034366775),
 (('link', 'rt'), 0.005083280283523205),
 (('social', 'network'), 0.004266154854770207),
 (('new', 'social'), 0.003829087764971923),
 (('googl', 'launch'), 0.003259000256539379),
 (('network', 'call'), 0.002964455043849231),
 (('call', 'circl'), 0.0028694404591104736),
 (('rt', 'googl'), 0.0027649244158978402),
 (('appl', 'open'), 0.002736420040476213),
 (('major', 'new'), 0.0027174171235284615),
 (('#sxsw', 'appl'), 0.002641405455737456),
 (('launch', 'major'), 0.002641405455737456),
 (('store', '#sxsw'), 0.00256539378794645),
 (('pop-up', 'store'), 0.002441874827786065),
 (('austin', '#sxsw'), 0.0024133704523644378),
 (('link', 'via'), 0.002375364618468935),
 (('today', 'link'), 0.002365863159995059),
 (('#sxsw', 'ipad'), 0.0022233412828869233),
 (('iphon', 'app'), 0.0022138398244130474),
 (('possibl', 'today'), 0.002194836907465296),
 (('#sxsw', 'googl'), 0.0021663325320436687),
 (('circl', 'possibl'), 0.0020998223227265385),
 (('rt', 'rt'), 0.0018527844024057692),
 (('temporari', 'store'), 0.0018242800269841142),
 (('#sxsw', '#sxswi'), 0.001748268359193136),
 (('store', 'austin'), 0.0017292654422453846),
 (('downtown', 'austin'), 0.0017007610668237575),
 (('googl', 'map'), 0.0016912596083498816),
 (('store', 'downtown'), 0.001643752315980503),
 (('\x89', 'ûï'), 0.0016342508575066272),
 (('\x89', 'û'), 0.0015772421066633728),
 (('rt', '#sxsw'), 0.0015487377312417457),
 (('marissa', 'mayer'), 0.0015202333558201183),
 (('googl', 'circl'), 0.0014727260634507396),
 (('open', 'temporari'), 0.001463224604976864),
 (('2', 'launch'), 0.0014347202295552368),
 (('ipad', 'app'), 0.0014157173126074852),
 (('rt', 'appl'), 0.0014062158541336095),
 (('û', '\x9d'), 0.0014062158541336095),
 (('2', '#sxsw'), 0.001282696893973225),
 (('sxsw', 'link'), 0.0012256881431299705),
 (('open', 'pop-up'), 0.0012066852261822189),
 (('new', 'ipad'), 0.0011876823092344675),
 (('...', 'link'), 0.0011781808507605918),
 (('#sxsw', '\x89'), 0.0010736648075479586),
 (('launch', 'link'), 0.0010736648075479586)]
```

Reviewing user sentiments on specific products and brands

A function that takes in a specific Product and analyzes sentiments

In [33]:
```
def analyze_sentiments_product(data, brand, emotion, top_n):
    # Filter the dataset for the specified brand and emotion
    brand_data = data[(data['Product/Brand'] == brand) & (data['Emotion'] == emotion)]

    # Concatenate all preprocessed text into a single list
    brand_tokens = [token for sublist in brand_data['preprocessed_text'] for token in sublist]

    # Create a frequency distribution of tokens
    brand_freq_dist = FreqDist(brand_tokens)

    # Get the most common tokens
    top_tokens = brand_freq_dist.most_common(top_n)

    # Extract the tokens and frequencies
    tokens, frequencies = zip(*top_tokens)

    # Plot the frequency distribution
    plt.figure(figsize=(10, 6))
    plt.bar(tokens, frequencies)
    plt.xlabel('Tokens')
    plt.ylabel('Frequency')
    plt.title(f'Top {top_n} {emotion.capitalize()} {brand} {"" if top_n == 1 else "s"}')
    plt.xticks(rotation=90)
    plt.show()
```

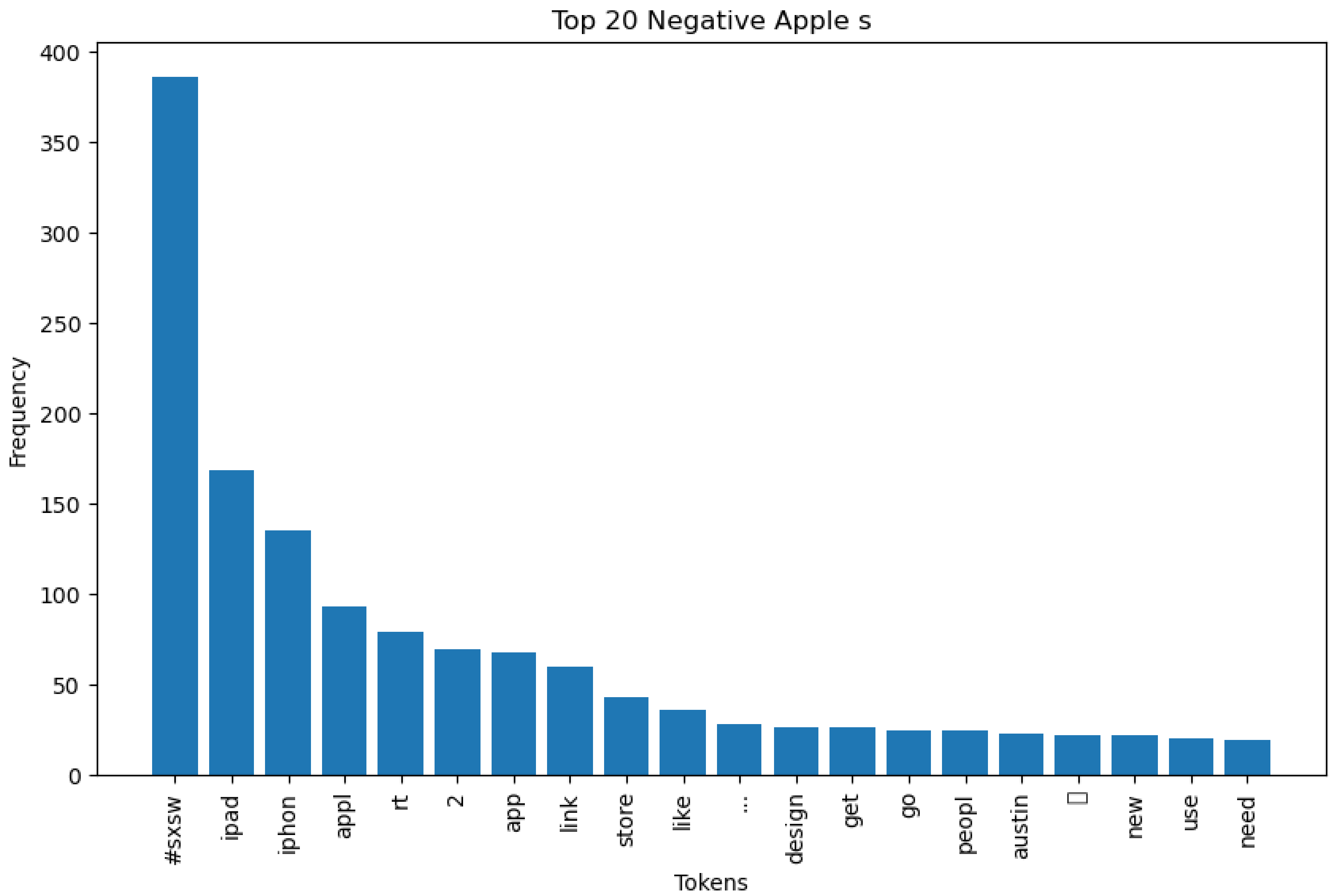
A function that takes the Brand as an argument and analyzes the sentiments

In [34]:
```
def analyze_sentiments(data, brand, emotion, top_n):
    # Filter the dataset for the specified brand and emotion
    brand_data = data[(data['Brand'] == brand) & (data['Emotion'] == emotion)]

    # Concatenate all preprocessed text into a single list
    brand_tokens = [token for sublist in brand_data['preprocessed_text'] for token in sublist]

    # Create a frequency distribution of tokens
    brand_freq_dist = FreqDist(brand_tokens)

    # Get the most common tokens
    top_tokens = brand_freq_dist.most_common(top_n)

    # Extract the tokens and frequencies
    tokens, frequencies = zip(*top_tokens)

    # Plot the frequency distribution
    plt.figure(figsize=(10, 6))
    plt.bar(tokens, frequencies)
    plt.xlabel('Tokens')
    plt.ylabel('Frequency')
    plt.title(f'Top {top_n} {emotion.capitalize()} {brand} {"" if top_n == 1 else "s"}')
    plt.xticks(rotation=90)
    plt.show()
```

Sentiments on the Apple Brand

1. Complaints

In [35]:
```
analyze_sentiments(data, "Apple", "Negative", 20)
```

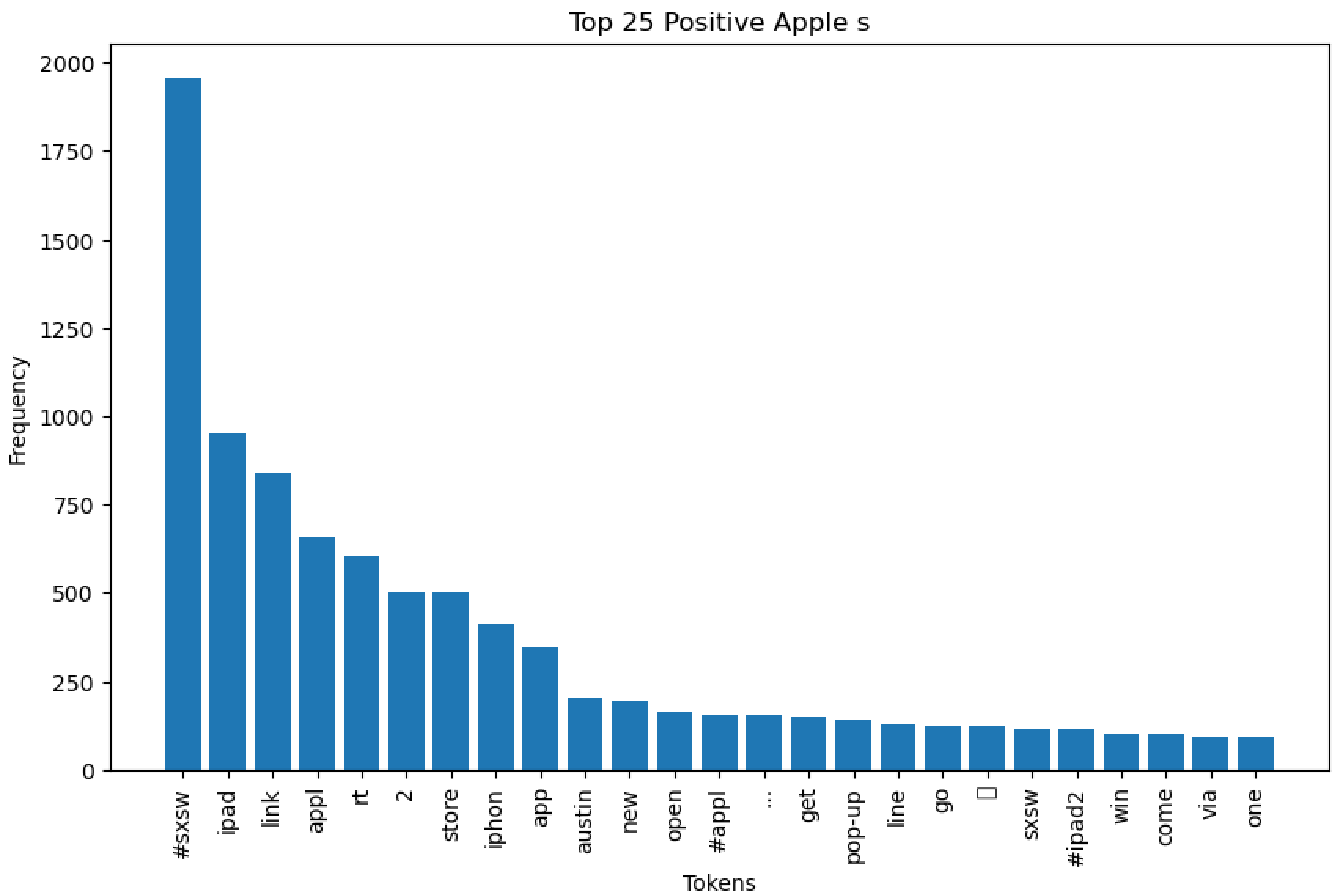


Noteworthy words used in apple product complaints are:

- iPhone
- design

2. Praises

In [36]:
```
analyze_sentiments(data, "Apple", "Positive", 25)
```



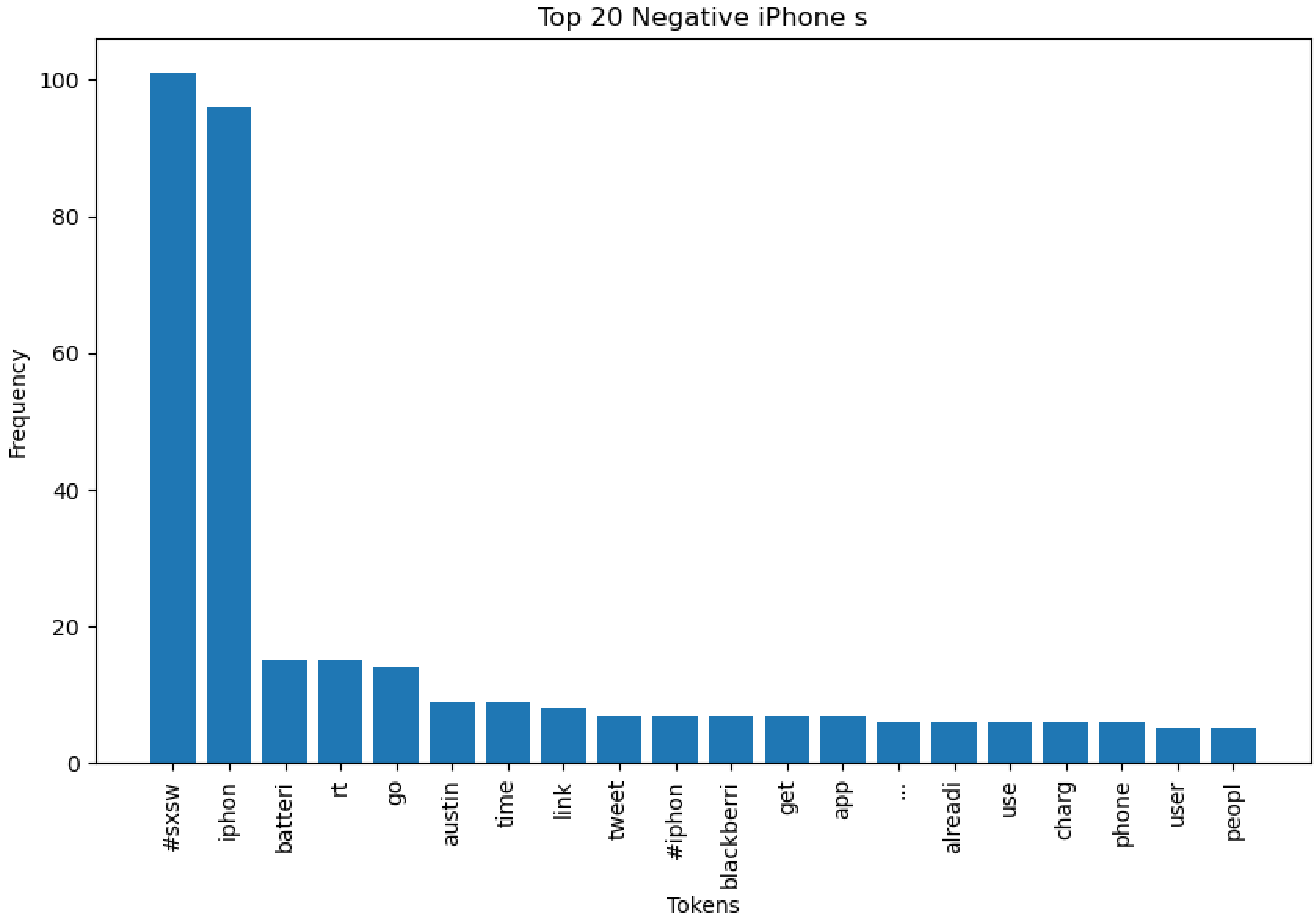
Noteworthy words used in apple product praises are:

- ipad

- ipad 2

Because from this analysis, the iphone and ipad are the most popular products, the sentiments on apple products were analyzed, focussing on these products. The following charts show iphone praises and complaints respectively:

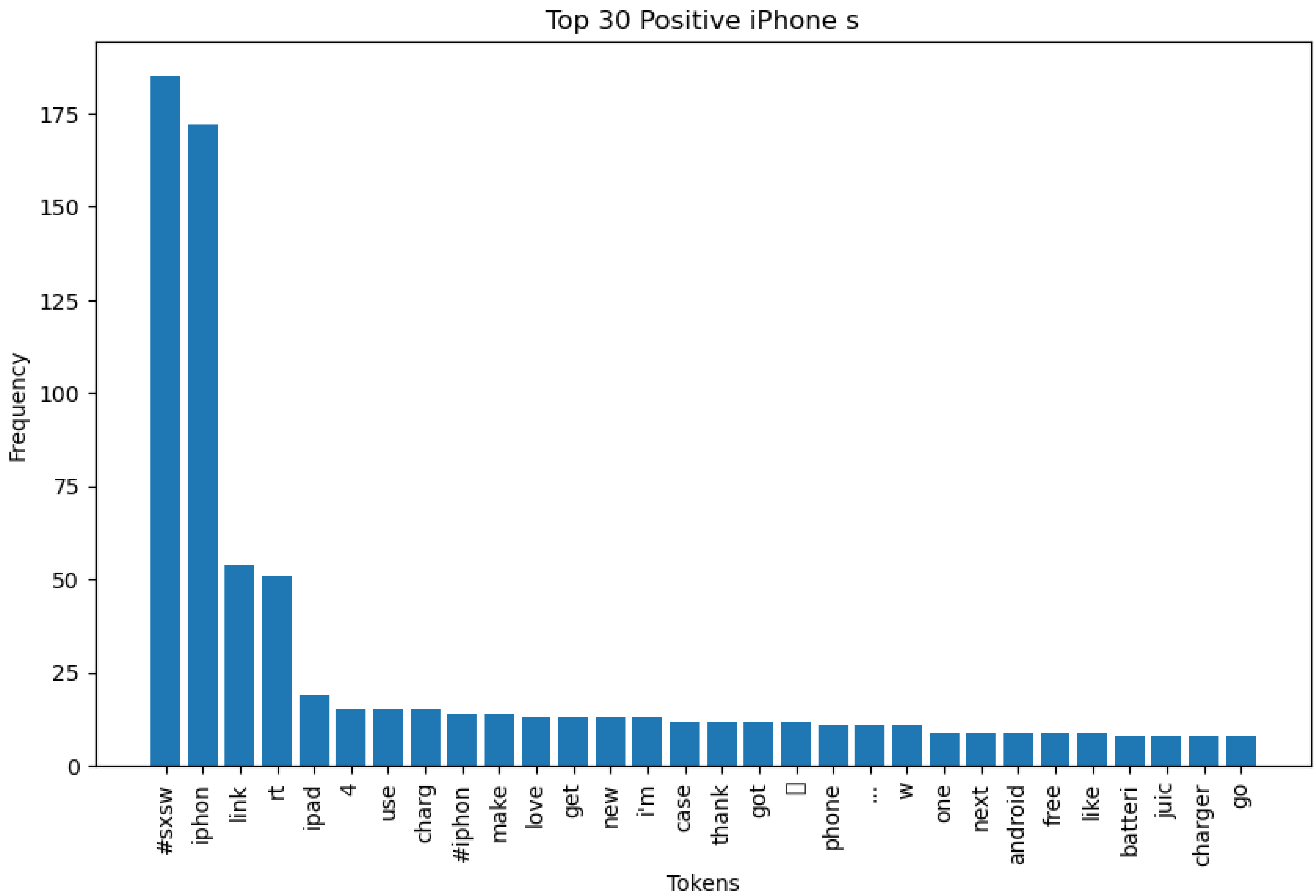
In [37]:
```
analyze_sentiments_product(data, 'iPhone', 'Negative', 20)
```



Noteworthy words used in iphone complaints are:

- Battery
- app
- charge

In [38]:
```
analyze_sentiments_product(data, 'iPhone', 'Positive', 30)
```



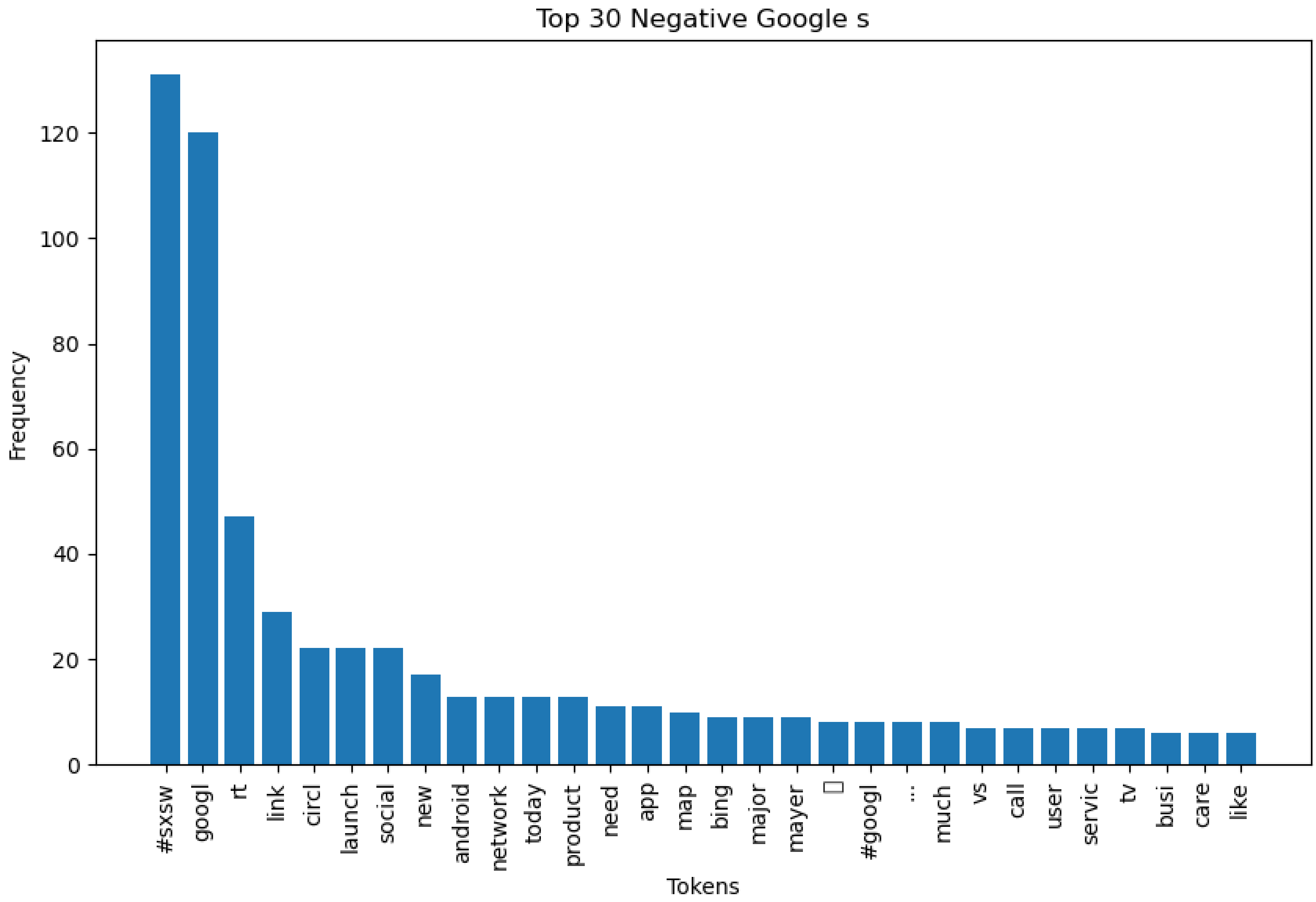
Noteworthy words used to praise the iphone are:

- pop-up
- Case
- Charger

Sentiments on the Google Brand

1. Complaints

In [39]:
```
analyze_sentiments(data, "Google", "Negative", 30)
```

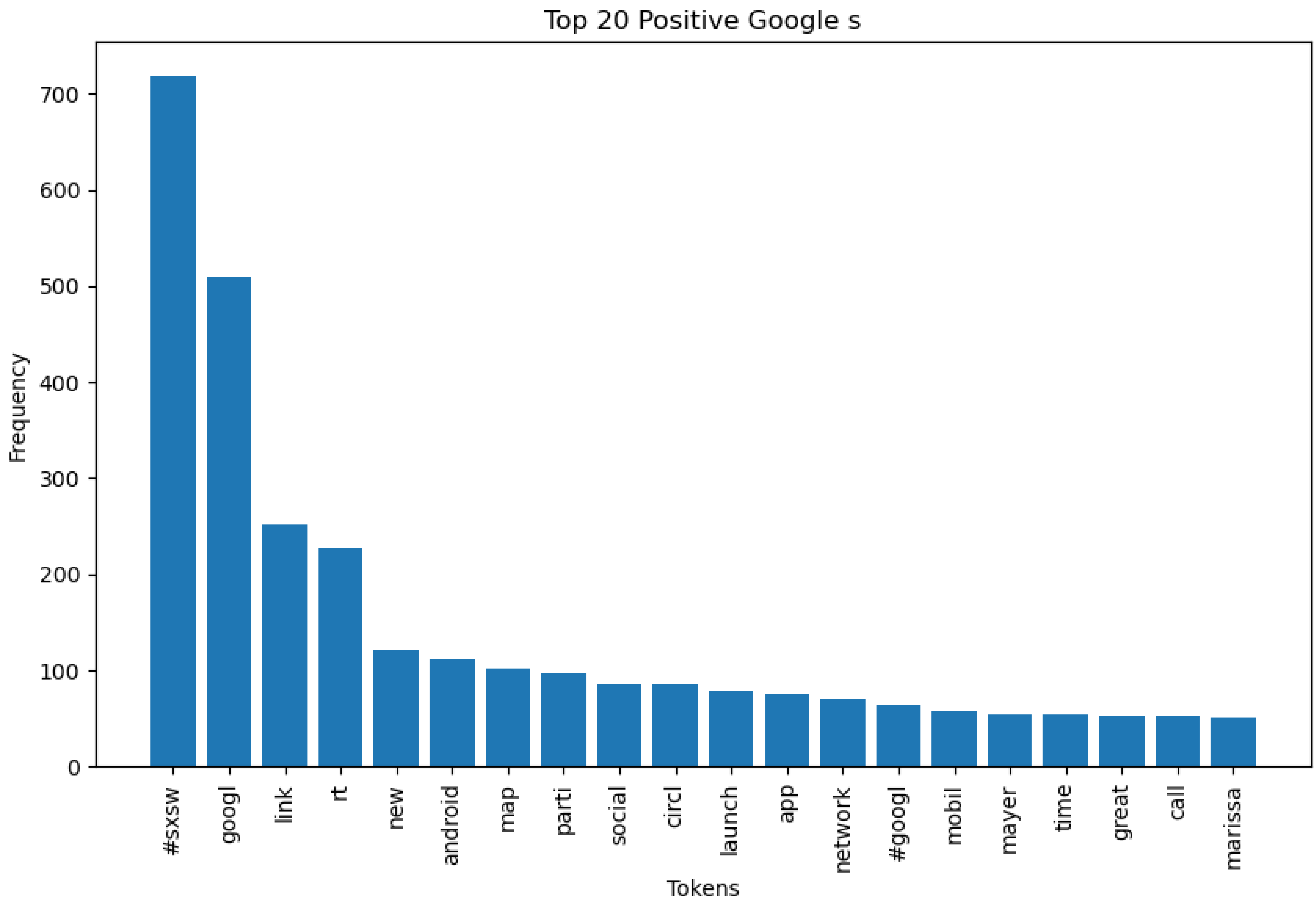


Noteworthy words used in google product complaints are:

- 'android'
- 'service'

2. Praises

In [40]:
```
analyze_sentiments(data, "Google", "Positive", 20)
```



Noteworthy words used in Google product praises are:

- Circle
- map

A function that takes in noteworthy words and produces a sample of tweets that contain that word for review

In [41]:
```
def get_tweets_with_word(data, word, sample_size=10):
    # Filter the dataset to include only tweets containing the specified word
    filtered_tweets = data[data['preprocessed_text'].apply(lambda x: word in x)]

    # Randomly sample tweets from the filtered dataset
    sampled_tweets = filtered_tweets.sample(n=sample_size)

    # Return the sampled tweets
    return sampled_tweets['Tweet']
```

In [42]:
```
word = 'circl'  # The word you want to search for
sample_size = 15  # Number of tweets to retrieve

sampled_tweets = get_tweets_with_word(data, word, sample_size)

# Print the sampled tweets
for tweet in sampled_tweets:
    print(tweet)
    print('---')
```

```
@mention - False Alarm: Google Circles Not Coming Now  ÛÒand Probably Not Ever? - {link} #Google #Circles #Soc
ial #SXSW
---
New circle game? RT @mention  @mention Google (tries again) to launch new social network called Circles: {lin
k} #sxsw
---
RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today @mention #sxsw #sxswi -
{link} via @mention
---
Those of you at #SXSW I need the details on Google Circle! What's it all about? @mention @mention
---
  ÛÏ@mention Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw  Û   #fb
---
C'mon already - @mention to Launch Major New Social Network Called Circles, Possibly Today by @mention {link}
via @mention #SxSW
---
RT @mention RT @mention Google to Launch Major New Social Network Called Circles, Possibly Today @mention #SXS
W {link} #privacy ?
---
RT @mention RT @mention Google to Launch Major New Social Network Called Circles, {link} #sxsw #nptech
---
#Google Circles - the search engine's social network to be released soon? #SXSW seem to think so. My blog: {li
nk} #socialmedia
---
New social network may debut at #SXSW Google Circles {link} #EatDrinkTweet  Not done with you yet!
---
Lots of chatter around Google's new social network, Circles #SXSW -KEK
---
So we get to see google fail at social on another day RT @mention Okay, no Google Circles debuting at #sxsw to
day
---
RT @mention Man, Google *should* launch Circles at #SXSW. Talk about striking while the iron is hot. | @mentio
n @mention
---
Watch out FB 4 big brother RT@mention Google to Launch Major New Social Network Called Circles, Possibly Today
{link} #sxsw
---
RT @mention Google will launch major new social network called Circles. #SXSW
---
```

In [43]:
```
word = 'map'  # The word you want to search for
sample_size = 15  # Number of tweets to retrieve

sampled_tweets = get_tweets_with_word(data, word, sample_size)

# Print the sampled tweets
for tweet in sampled_tweets:
    print(tweet)
    print('---')
```

```
@mention Tempted? RT @mention Come party down @mention &amp; Google tonight #sxsw: {link} Bands, food, art, in
teractive maps
---
Google Maps Street View car sighting!!! #SXSW   {link}
---
love google street maps. Getting an idea of where everything is, in relation to the Austin Convention Center.
Alamo Drafthouse Cinema #sxsw
---
RT @mention Interesting Google Mobile Stats at #SXSW - 40% of all Google Maps users (150 million) come from #m
obile {link}
---
40% of google map search is mobile #sxsw (@mention ACC - Ballroom D w/ 78 others) {link}
---
Going to #SXSW? Here's a google map of free wi-fi hotspots: {link} (warning: it's from 2009)
---
Google Marissa Mayer: mobile phone as a cursor of physical location - new version of map fast and more real li
fe like   #sxsw
---
Google Maps now has 150 million users on mobile #sxsw (via @mention
---
Mayer Maps Out Google's Coming Location Dominance at #SXSW {link} via @mention
---
Google showcase peak view of new Google Maps (vector-based maps) at #sxsw #marissagoogle
---
Wow! Google maps for mobile v5 demo at #sxsw. Very nice.
---
Marissa Mayer at #SXSW: 150 M users of Google Maps for mobile. For 1st time ever, an app born for web has more
use as a mobile app.
---
RT @mention Wanna know where the fast cellular signal is at #SXSW? Get Coverage: Austin for iOS (FREE) for det
ailed maps - {link}
---
RT @mention RT @mention For everyone at #SXSW here is a map of my favorite places in Austin {link} @mention @m
entione
---
Google Maps mobile is going 3D in the next release, looks awesome.  #sxsw
---
```

Summary of customer sentiments for Google and apple products :

Praises of `Apple Products`:

- The ipad 2 was launched and customers seemed to enjoy the improvement in design made over the ipad.
- The new iphone cases released during the SXSW Conference recieved a lot of attention on the tweets and the comments were generally very positive.
- There was a tweet thread of 'iphone vs android' and iphone was the more popular choice among users.
- The pop-up store in Austin created a lot of attention on the tweets and users in that area generally enjoyed the experience there.

Complaints of `Apple Products`:

- The battery life of the iphone was a major source of complaint. Users complained that it does not last a whole day.
- The apple music app was described by some users as "one of the worst apps I have had to use in a long time"
- The increase in size of the new ipad 2 was disapproved by some users as 'too big'

Praises of `Google Products`:

- The PacMan app on Android Opertaing system was praised.
- Users were very interested in Google maps and it proved to be the most popular Maps application with Marrissa Mayer stating that, "Usage of google maps surpused online use in the past couple of months" and that there were 150 million google map users.
- Google chrome browser had positive reviews when compared to Windows explorer with one user explaining that, "The switch is done immediately they buy a new laptop"
- A new Circle social media app the was rumoured to be launched created a lot of excitement among twitter users ahead of the launch event

Complaints of `Google Products`:

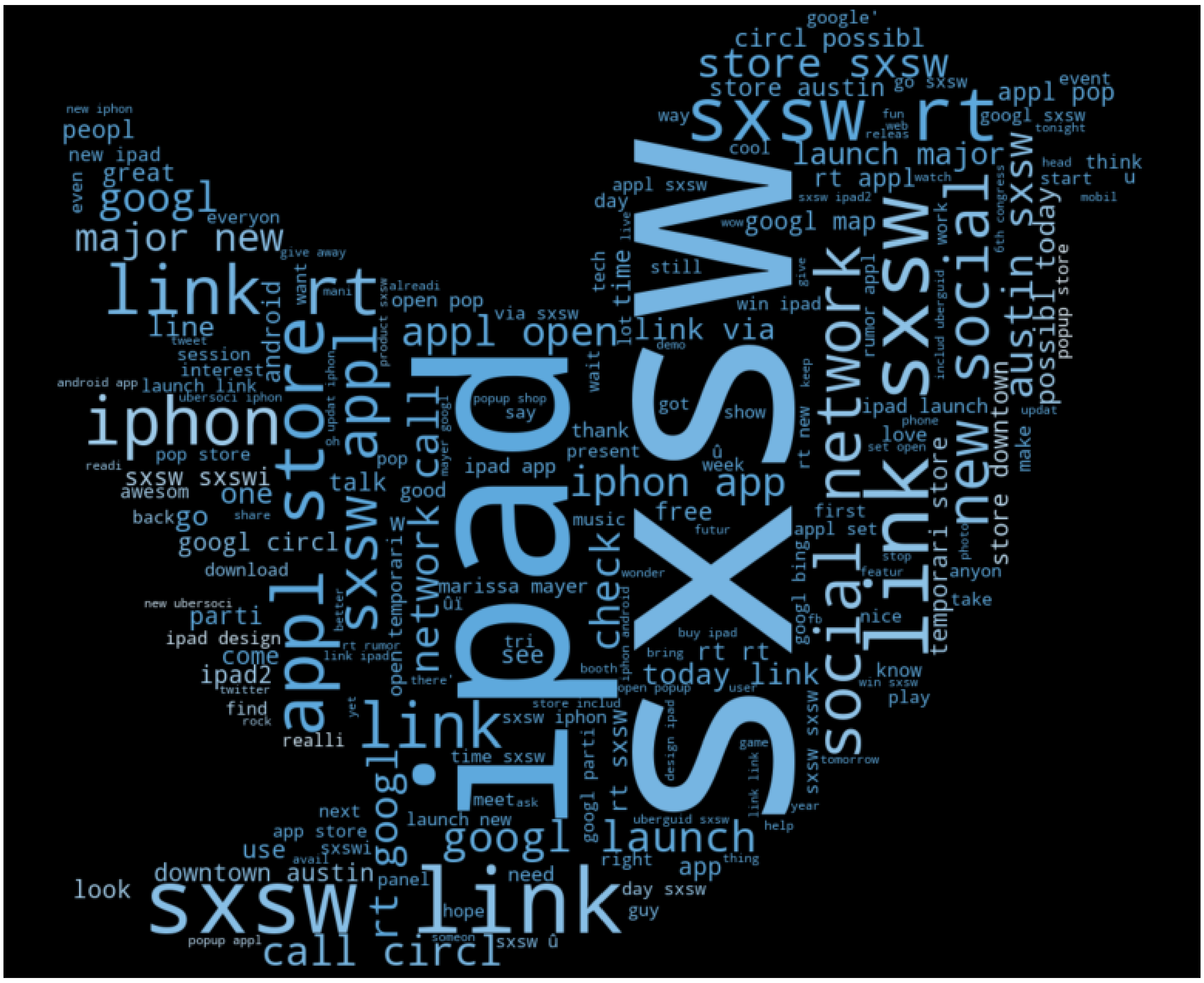
- The Android operating system was the biggest pain point for most users who tweeted with bugs being the major source of concern, one user describing the experience as 'Painful'
- Samsung products have been mentioned with the Android discussion and users stated that its implementation of android was better than that in google devices.
- Customer service at google was an issue with users complaining of a lack of refunds when returning faulty devices e.g the Nexus smartphone

WordCloud

Plotted the most tweeted words into a word cloud.

In [44]:
```
all_words_positive = ' '.join(' '.join(text) for text in data['preprocessed_text'])
```

In [45]:
```
# combining the image with the dataset
Mask = np.array(Image.open(requests.get('http://clipart-library.com/image_gallery2/Twitter-PNG-Image.png', strea

# We use the ImageColorGenerator library from Wordcloud
# Here we take the color of the image and impose it over our wordcloud
image_colors = ImageColorGenerator(Mask)

# Now we use the WordCloud function from the wordcloud library
wc = WordCloud(background_color='black', height=1500, width=4000,mask=Mask).generate(all_words_positive)

# Size of the image generated
plt.figure(figsize=(10,20))

# Here we recolor the words from the dataset to the image's color
# recolor just recolors the default colors to the image's blue color
# interpolation is used to smooth the image generated
plt.imshow(wc.recolor(color_func=image_colors),interpolation="hamming")

plt.axis('off')
plt.show()
```



6. Modelling

In [180]:
```
# Binary dataframe
data_binary = data.loc[(data['Emotion'] == 'Positive' ) | (data['Emotion'] == 'Negative' )]
data_binary
```

Out[180]:

	Tweet	Product/Brand	Emotion	preprocessed_text	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	Negative	[3g, iphon, 3, hr, tweet, #rise_austin, dead, need, upgrad, plugin, station, #sxsw]	Apple
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	[know, awesom, ipad, iphon, app, like, appreci, design, also, they'r, give, free, ts, #sxsw]	Apple
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	iPad	Positive	[wait, #ipad, 2, also, sale, #sxsw]	Apple
3	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	iPad or iPhone App	Negative	[hope, year', festiv, crashi, year', iphon, app, #sxsw]	Apple
4	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	Google	Positive	[great, stuff, fri, #sxsw, marissa, mayer, googl, tim, o'reilli, tech, book, confer, matt, mullenweg, wordpress]	Google
...	...	...	...	...	...
9077	@mention your PR guy just convinced me to switch back to iPhone. Great #sxsw coverage. #princess	iPhone	Positive	[pr, guy, convinc, switch, back, iphon, great, #sxsw, coverag, #princess]	Apple
9079	"papyrus...sort of like the ipad" - nice! Lol! #SXSW Lavelle	iPad	Positive	[papyru, ..., sort, like, ipad, nice, lol, #sxsw, lavel]	Apple
9080	Diller says Google TV "might be run over by the PlayStation and the Xbox, which are essentially ready today." #sxsw #diller	Other Google product or service	Negative	[diller, say, googl, tv, might, run, playstat, xbox, essenti, readi, today, #sxsw, #diller]	Google
9085	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	iPad or iPhone App	Positive	[i'v, alway, use, camera, iphon, b, c, imag, stabil, mode, suggest, ipad, cam, app, w, featur, #sxsw, #sxswi]	Apple
9088	Ipad everywhere. #SXSW {link}	iPad	Positive	[ipad, everywher, #sxsw, link]	Apple

3539 rows × 5 columns

In [181]:
```
# Relabeling the content of the emotion column
data_binary.loc[data_binary['Emotion'] == 'Positive' , 'emotion_label'] = 0
data_binary.loc[data_binary['Emotion'] == 'Negative' , 'emotion_label'] = 1
data_binary
```

Out[181]:

	Tweet	Product/Brand	Emotion	preprocessed_text	Brand	emotion_label
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	[3g, iphon, 3, hr, tweet, #rise_austin, dead, need, upgrad, plugin, station, #sxsw]	Apple	1.0
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	[know, awesom, ipad, iphon, app, like, appreci, design, also, they'r, give, free, ts, #sxsw]	Apple	0.0
2	@swonderlin Can not wait for #iPad 2 also. They should sale them down at #SXSW.	iPad	Positive	[wait, #ipad, 2, also, sale, #sxsw]	Apple	0.0
3	@sxsw I hope this year's festival isn't as crashy as this year's iPhone app. #sxsw	iPad or iPhone App	Negative	[hope, year', festiv, crashi, year', iphon, app, #sxsw]	Apple	1.0
4	@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences) & Matt Mullenweg (Wordpress)	Google	Positive	[great, stuff, fri, #sxsw, marissa, mayer, googl, tim, o'reilli, tech, book, confer, matt, mullenweg, wordpress]	Google	0.0
...	...	...	...	...	...	...
9077	@mention your PR guy just convinced me to switch back to iPhone. Great #sxsw coverage. #princess	iPhone	Positive	[pr, guy, convinc, switch, back, iphon, great, #sxsw, coverag, #princess]	Apple	0.0
9079	"papyrus...sort of like the ipad" - nice! Lol! #SXSW Lavelle	iPad	Positive	[papyru, ..., sort, like, ipad, nice, lol, #sxsw, lavel]	Apple	0.0
9080	Diller says Google TV "might be run over by the PlayStation and the Xbox, which are essentially ready today." #sxsw #diller	Other Google product or service	Negative	[diller, say, googl, tv, might, run, playstat, xbox, essenti, readi, today, #sxsw, #diller]	Google	1.0
9085	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	iPad or iPhone App	Positive	[i'v, alway, use, camera, iphon, b, c, imag, stabil, mode, suggest, ipad, cam, app, w, featur, #sxsw, #sxswi]	Apple	0.0
9088	Ipad everywhere. #SXSW {link}	iPad	Positive	[ipad, everywher, #sxsw, link]	Apple	0.0

3539 rows × 6 columns

In [182]:
```
# Split the data -- Train and Test

from sklearn.model_selection import train_test_split
data['joined_preprocessed_text'] = data['preprocessed_text'].str.join(" ")
X_train, X_test, y_train, y_test = train_test_split(data['joined_preprocessed_text'], data.
                                                    Emotion, test_size = 0.3,
                                                     random_state=20)
X_train
```

Out[182]:
```
3397                    i10 march pig link code valid 12:00- 3:59 59p 03/12 11 #infektd #sxsw #zomb
4321      lol rt i'll bet there' lot nerd #sxsw use #iphon light saber app barroom brawl instead fist
4091                                  #appl ipad per capit #sxsw anywher els world except foxconn
6378            rt umbrella list via    ÷ ¼ set    ÷ link    ÷ #edchat #musedchat #sxsw #sxswi #newtwitt
5207             rt #appl head sxsw set temporari store #austin link #new #sxsw bastard bring temptat
                                                                     ...
6030                   rt hoot new blog post hootsuit mobil #sxsw updat iphon blackberri android link
3994                                    hannukah miracl morn uncharg iphon still 55 batteri #sxsw
7208             alarm go allow user gener content say google' richard salgado #privacybootcamp #sxsw
7540                                       head ipad design headach 2 tablet call morn #sxsw link
4454                                 sit floor behind guy who' fondl new ipad 2 disturb way #sxsw
Name: joined_preprocessed_text, Length: 6239, dtype: object
```

In [183]:
```
# Split the data -- Train and Test

from sklearn.model_selection import train_test_split
data_binary['joined_preprocessed_text'] = data_binary['preprocessed_text'].str.join(" ")
X_train, X_test, y_train, y_test = train_test_split(data_binary['joined_preprocessed_text'], data_binary.
                                                    Emotion, test_size = 0.3,
                                                     random_state=20)
X_train
```

Out[183]:
```
5377      rt agre    ûó novelti #ipad news #app fade fast among digit deleg    ûó link #media #sxsw via
6874                                                rt includ emot thing ipad #barrydil #sxsw #pnid
8683                   pick mophi iphon charg case #sxsw tradeshow doubt i'll use time great long day
6147                                                                rt ipad 2 queue epic #sxsw link
6759                      rt new whrrl app live iphon app store android marketplac get hot time #sxsw
                                                                     ...
7601                                                   dear lanyrd site awesom use #sxsw wish iphon app
8436                               saw huge ipad 2 line popup appl store peopl realli realli want #sxsw
626           wonder much revenu austin store gross account ipad 2 amid #sxsw ... strateg releas ...
6349                                   rt new #ubersoci #iphon app store includ uberguid #sxsw link got
8800                                                   protip avoid austin-area appl store friday #sxsw
Name: joined_preprocessed_text, Length: 2477, dtype: object
```

In [184]:
```
# Instantiate a vectorizer with max_features=10
# (we are using the default token pattern)
tfidf = TfidfVectorizer(max_features=10)

# Fit the vectorizer on X_train and transform it
X_train_vectorized = tfidf.fit_transform(X_train)

# Get the feature names from the vectorizer
feature_names = tfidf.get_feature_names()

# Visually inspect the vectorized data
pd.DataFrame.sparse.from_spmatrix(X_train_vectorized, columns=feature_names)
```

Out[184]:

	app	appl	googl	ipad	iphon	link	new	rt	store	sxsw
0	0.609007	0.000000	0.0	0.429649	0.000000	0.421569	0.000000	0.472228	0.000000	0.209265
1	0.000000	0.000000	0.0	0.639496	0.000000	0.000000	0.000000	0.702872	0.000000	0.311473
2	0.000000	0.000000	0.0	0.000000	0.934017	0.000000	0.000000	0.000000	0.000000	0.357229
3	0.000000	0.000000	0.0	0.541689	0.000000	0.531503	0.000000	0.595372	0.000000	0.263835
4	0.723758	0.000000	0.0	0.000000	0.325121	0.000000	0.398759	0.280603	0.342435	0.124347
...	...	...	...	...	...	...	...	...	...	...
2472	0.720705	0.000000	0.0	0.000000	0.647499	0.000000	0.000000	0.000000	0.000000	0.247646
2473	0.000000	0.544773	0.0	0.481253	0.000000	0.000000	0.000000	0.000000	0.645503	0.234399
2474	0.000000	0.000000	0.0	0.573889	0.000000	0.000000	0.000000	0.000000	0.769754	0.279518
2475	0.442144	0.000000	0.0	0.000000	0.397233	0.306063	0.487205	0.342842	0.418388	0.151928
2476	0.000000	0.621475	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.736387	0.267402

2477 rows × 10 columns

Baseline Model

In [185]:
```
#instantiate
baseline_model = MultinomialNB()

baseline_model.fit(X_train_vectorized,y_train)
baseline_train_score = baseline_model.score(X_train_vectorized,y_train)
print(baseline_train_score)
```

```
0.8364957610012111
```

In [186]:
```
# imbalanced
y_train.value_counts(normalize=True)
```

Out[186]:
```
Positive    0.836496
Negative    0.163504
Name: Emotion, dtype: float64
```

In [187]:
```
# Balancing the model
ROS = RandomOverSampler(sampling_strategy=1)
X_train_ros, y_train_ros = ROS.fit_resample(X_train_vectorized, y_train)
```

In [188]:
```
y_train_ros.value_counts(normalize=True)
```

Out[188]:
```
Negative    0.5
Positive    0.5
Name: Emotion, dtype: float64
```

Model 2

In [189]:
```

# Balancing the model
ROS = RandomOverSampler(sampling_strategy=1)
X_train_ros, y_train_ros = ROS.fit_resample(X_train_vectorized, y_train)
```

In [190]:
```
y_train_ros.value_counts(normalize=True)
```

Out[190]:
```
Negative    0.5
Positive    0.5
Name: Emotion, dtype: float64
```

In [191]:
```
model2 = MultinomialNB()

model2.fit(X_train_ros, y_train_ros)
train_score2 = model2.score(X_train_ros, y_train_ros)
print(train_score2)
```

```
0.6078667953667953
```

In [192]:
```

model2_train_predictions = model2.predict(X_train_ros)
model2_f1 = f1_score(y_train_ros, model2_train_predictions, pos_label='Positive')

print(model2_f1)
```

```
0.5775929295554977
```

In the binary classification section, we focused on optimizing our models for the F1 score since we would like our model to predict both negative and positive tweets correctly. For the binary classification problem, the best model was the tuned random oversampled Multinomial Naive Bayes model based on the score of 0.61 on the training dataset, and an F1 score of 0.58.

Multiclass Classification

After running a Binary classification we will also look at Multiclass Classification to see how it performs on the data. We will start with preparing data for multiclass

In [193]:
```
# After running a Binary classification we will also look at Multiclass Classification
# see how it performs on the data.
# We will start with preparing data for multiclass
# Che the data column of Emotion
data['Emotion'].unique()
```

Out[193]:
```
array(['Negative', 'Positive', 'Neutral'], dtype=object)
```

In [194]:
```
# We will assign numerical values to the sentiments,
# Neutral = 1
# Positive = 2
# Negarive = 0
#mapping emotion column to numerical values
emotion_dict = {'Negative': 0, 'Neutral':1, 'Positive': 2}
data['Emotion'] = data['Emotion'].map(emotion_dict)
data['Emotion'].value_counts()
```

Out[194]:
```
1    5375
2    2970
0     569
Name: Emotion, dtype: int64
```

In [195]:
```
# Define the X and y objects
X = data['Tweet']
y = data['Emotion']

# split into train and test objects
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=42)

# check the y_train values
y_train.value_counts(normalize=True)
```

Out[195]:
```
1    0.602721
2    0.333894
0    0.063385
Name: Emotion, dtype: float64
```

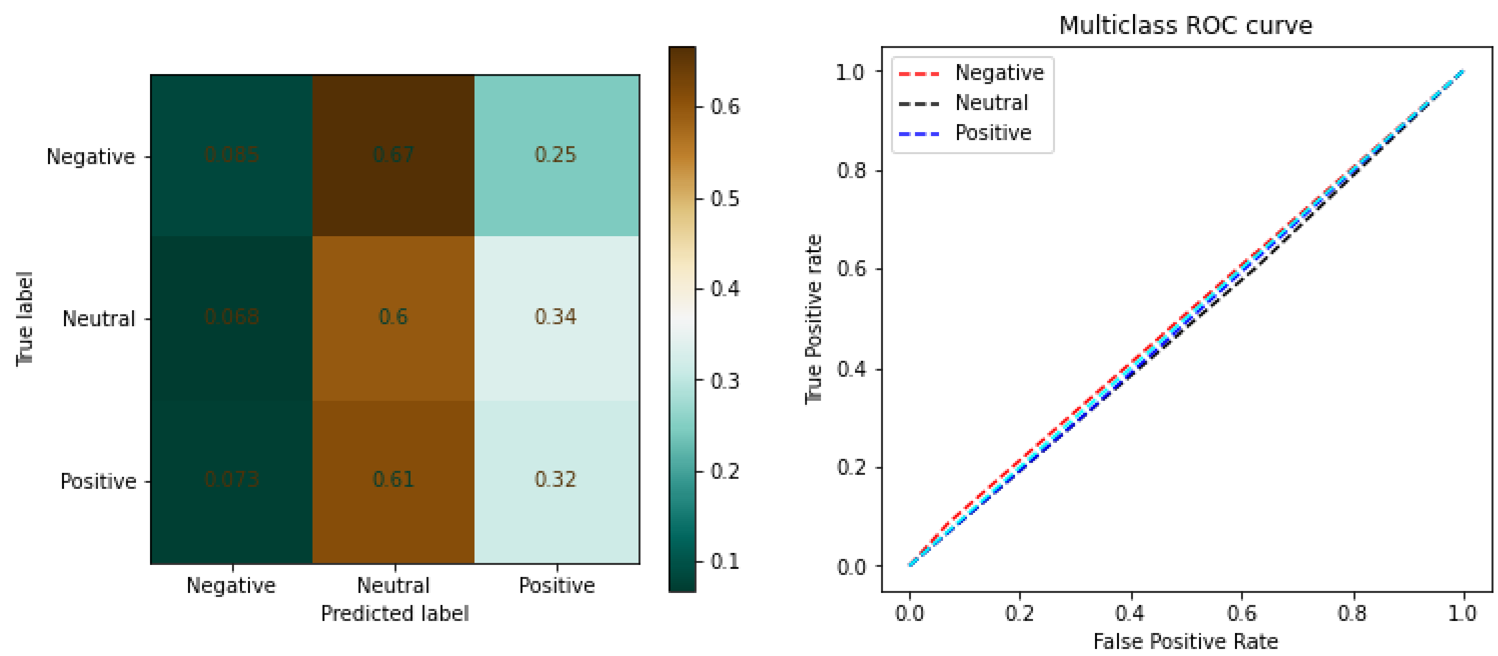
In [196]:
```
g a multiclass classification that take in y_true as the vlues of y_test, y_pred, initial classifier, X_test and

class_eval(y_true, y_pred, X_test, X_train, clf, n_class=3):
(f"Training score: {clf.score(X_train, y_train)}\
est Score:{clf.score(X_test, y_test)}")

nt classication report
('\n')
('Classifictaion Report')
('________________________________________')
(classification_report(y_true=y_true, y_pred=y_pred))

ate a figure/axes for confusion matrix and ROC curve
ax = plt.subplots(ncols=2, figsize=(12, 5))

t the normalized confusion matrix
confusion_matrix(estimator=clf, X=X_test, y_true=y_true, cmap='BrBG_r',
                 normalize='true', ax=ax[0],
                 display_labels=['Negative', 'Neutral', 'Positive'])

prob = clf.predict_proba(X_test)

t the ROC curve
}
}
h={}

 in range(n_class):
pr[i], tpr[i], thresh[i] = roc_curve(y_true, pred_prob[:,i], pos_label=i)

.plot(fpr[0], tpr[0], linestyle='--',color='red', label='Negative')
.plot(fpr[1], tpr[1], linestyle='--',color='black', label='Neutral')
.plot(fpr[2], tpr[2], linestyle='--',color='blue', label='Positive')
.set_title('Multiclass ROC curve')
.set_xlabel('False Positive Rate')
.set_ylabel('True Positive rate')
.legend(loc='best')


.plot([0,1], [0,1], ls='--', color='cyan')
```

1. Dummy Model

In [197]:

```python
# Create a Dummy model for Multiclass

tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True)

dm_pipe = Pipeline([
    ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                   stop_words=stopwords_list)),
    ('clf', DummyClassifier(random_state=42))
])

dm_pipe.fit(X_train, y_train)
y_pred = clf_pipe.predict(X_test)
multiclass_eval(y_test, y_pred, X_test, X_train, dm_pipe)
```

```
Training score: 0.48548590660496427        Test Score:0.4699943914750421


Classifictaion Report
_______________________________________
              precision    recall  f1-score   support

           0       0.15      0.70      0.25       117
           1       0.77      0.56      0.65      1077
           2       0.57      0.43      0.49       589

    accuracy                           0.53      1783
   macro avg       0.50      0.56      0.46      1783
weighted avg       0.66      0.53      0.57      1783
```



The above outcome show class imbalance and that is why we have more neutrals comming out as the majority followed by positives and then negatives.

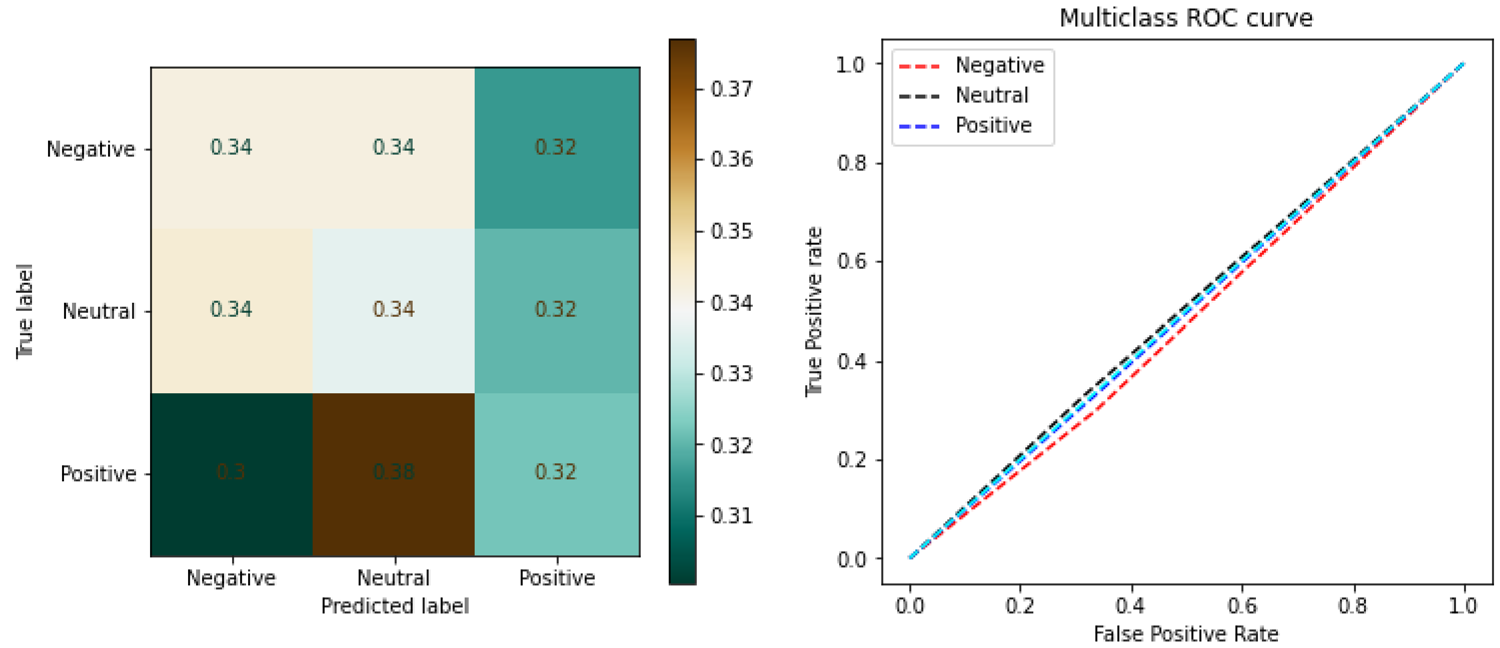
This reflects back to the our classes on the value counts in the `Emotion` column. Class imbalance can affect our model by being biased towords the majority in the values. Meaning any time we run our model it has 60% chances of selecting a neutral sentiment, 33% chances of selecting a positive sentiment and 6% chances of selecting a negative sentiment.

Class imbalance can be addressed by `Oversampling` method. In the next cell we will do random oversampling and then split the data and run a model to see how it will perform.

In [198]:

```python
# Random Classifier

rdm_pipe = imblearn.pipeline.Pipeline([
    ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                   stop_words=stopwords_list)),
    ('os', RandomOverSampler(random_state=42)),
    ('clf', DummyClassifier())
])
rdm_pipe.fit(X_train, y_train)
y_pred = clf_pipe.predict(X_test)
multiclass_eval(y_test, y_pred, X_test, X_train, rdm_pipe)
```

```
Training score: 0.3355770579161408        Test Score:0.34380257992148067


Classifictaion Report
_______________________________________
              precision    recall  f1-score   support

           0       0.15      0.70      0.25       117
           1       0.77      0.56      0.65      1077
           2       0.57      0.43      0.49       589

    accuracy                           0.53      1783
   macro avg       0.50      0.56      0.46      1783
weighted avg       0.66      0.53      0.57      1783
```

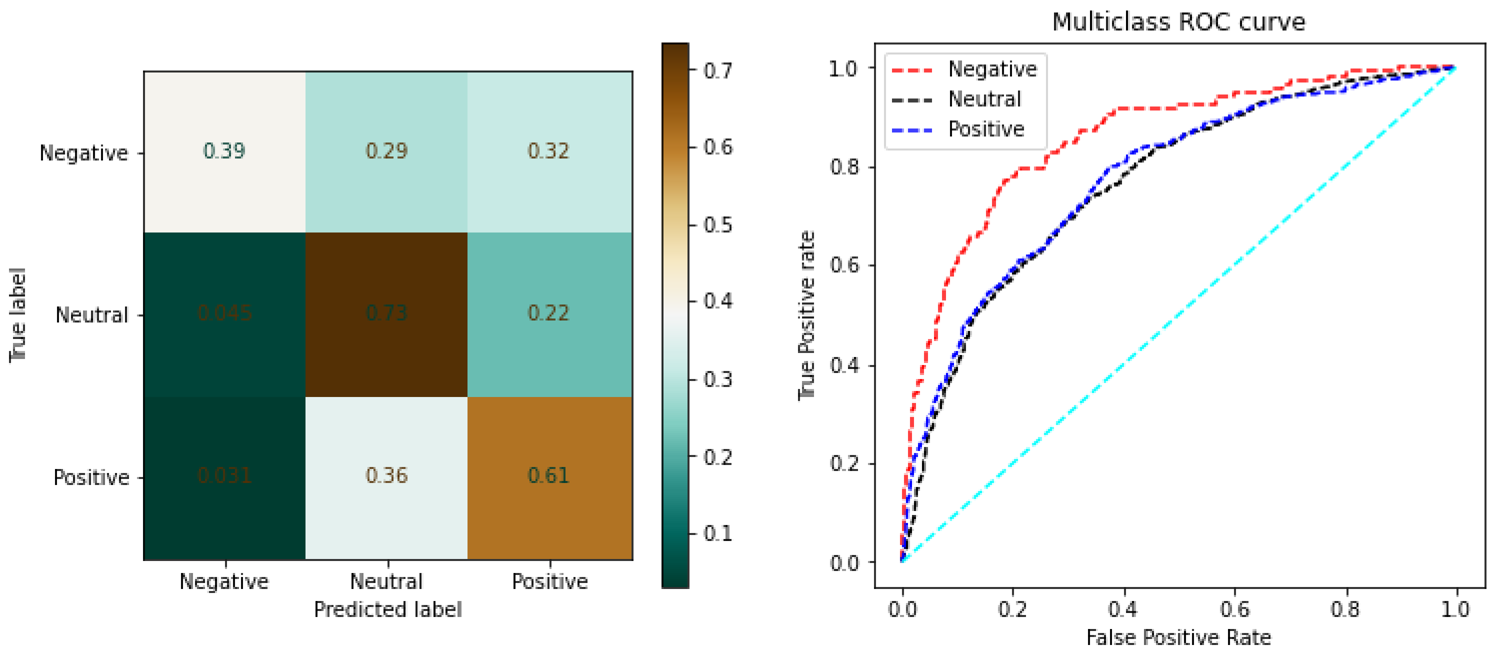


From the observation above it looks like the model is now able to levelup the sentiments and any sentment has equal chances to be selected.

2. Logistic Regression.

In [199]:
```
# Create a pipeline baseline logistic Regression
clf_pipe = Pipeline([
    ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                   stop_words=stopwords_list)),
    ('clf', LogisticRegressionCV(class_weight='balanced', random_state=42,
                                 cv=3))
])

clf_pipe.fit(X_train, y_train)
y_pred = clf_pipe.predict(X_test)
multiclass_eval(y_test, y_pred, X_test, X_train, clf_pipe)
```

```
Training score: 0.9389987379049222        Test Score:0.6713404374649468


Classifictaion Report
_______________________________________
              precision    recall  f1-score   support

           0       0.41      0.39      0.40       117
           1       0.76      0.73      0.75      1077
           2       0.57      0.61      0.59       589

    accuracy                           0.67      1783
   macro avg       0.58      0.58      0.58      1783
weighted avg       0.68      0.67      0.67      1783
```



Without tuning our model was able to classifie about 67% on the on the unseen data. This is a better model compaired to the dummy model, although there is some sort of overfitting the training data. Next we will look for the way to reduce the overfitting by the use of GridSearchCV.

Hyperparameter Tuning with GridSearchCV.

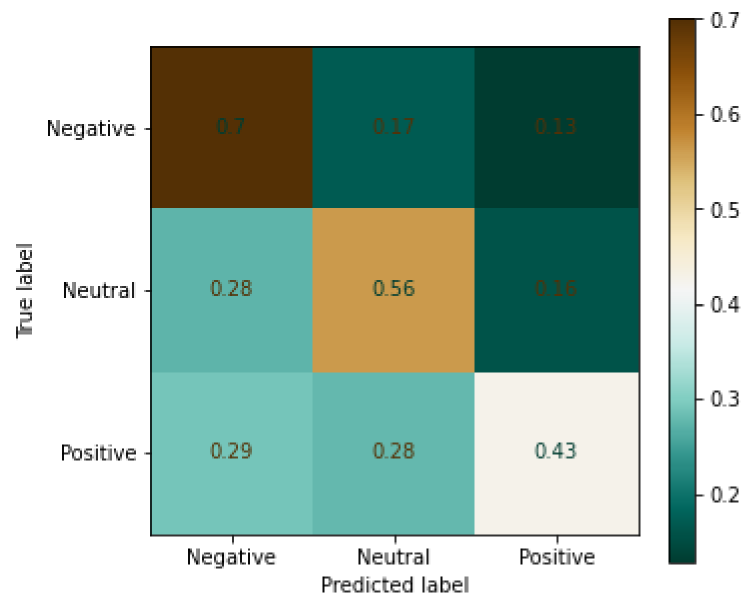
In [200]:
```
# Greating a GridSearchCV

param_grid = {
    'clf__class_weight': ['balanced'],
    'clf__max_iter': [2, 5],
    'clf__Cs': [0.1, 1],
    'clf__solver': ['liblinear', 'lbfgs', 'sag']
}

gs_lr = GridSearchCV(estimator=clf_pipe, param_grid=param_grid, scoring='recall_macro')
gs_lr.fit(X_train, y_train)
gs_lr.best_params_
```

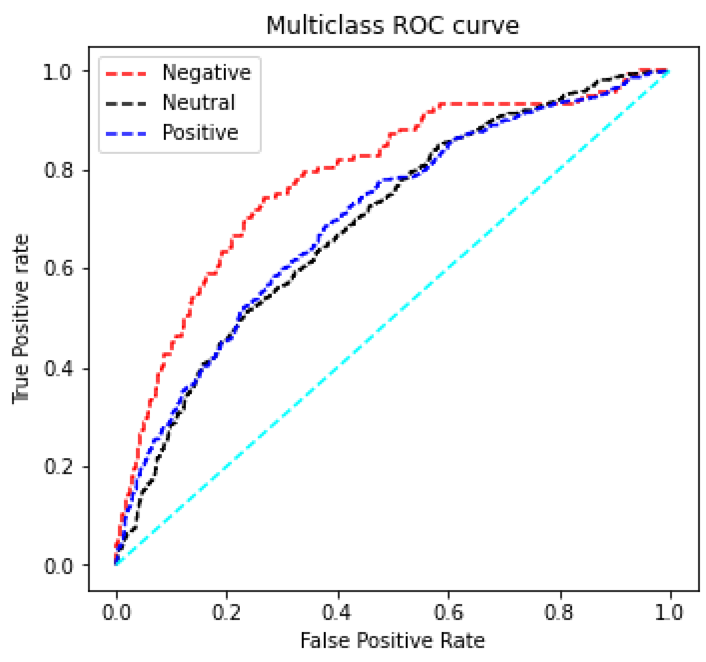
Out[200]:
```
{'clf__Cs': 1,
 'clf__class_weight': 'balanced',
 'clf__max_iter': 2,
 'clf__solver': 'lbfgs'}
```

In [201]:
```
# Creating a pipeline model for Logistic Resgression
gs_clf_pipe = Pipeline([
    ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                   stop_words=stopwords_list)),
    ('clf', LogisticRegressionCV(class_weight='balanced', Cs = [1],
                                 random_state=42, max_iter=2,
                                 solver='lbfgs',
                                 cv=3))
])

gs_clf_pipe.fit(X_train, y_train)
y_pred = clf_pipe.predict(X_test)
multiclass_eval(y_test, y_pred, X_test, X_train, gs_clf_pipe)
```

```
Training score: 0.5352685457860048        Test Score:0.5272013460459899


Classifictaion Report
_______________________________________
```

	precision	recall	f1-score	support
0	0.41	0.38	0.39	117
1	0.76	0.73	0.75	1077
2	0.57	0.61	0.59	589
accuracy			0.67	1783
macro avg	0.58	0.58	0.58	1783
weighted avg	0.67	0.67	0.67	1783



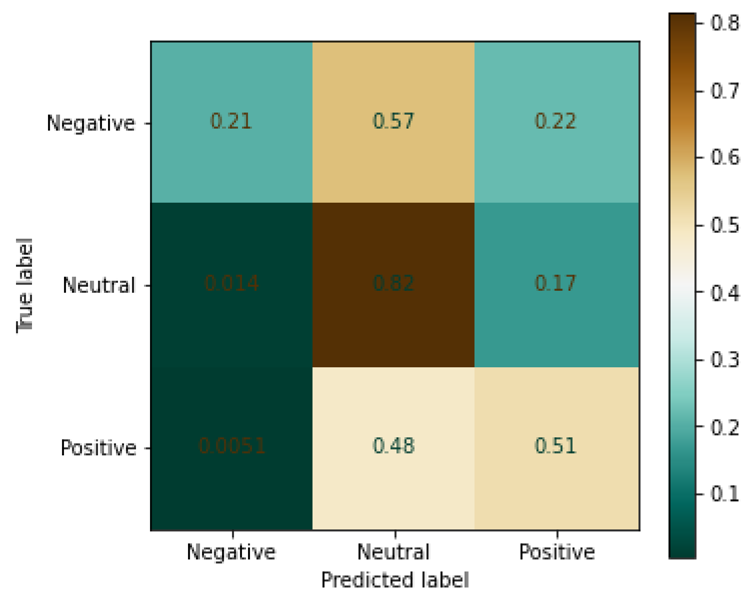


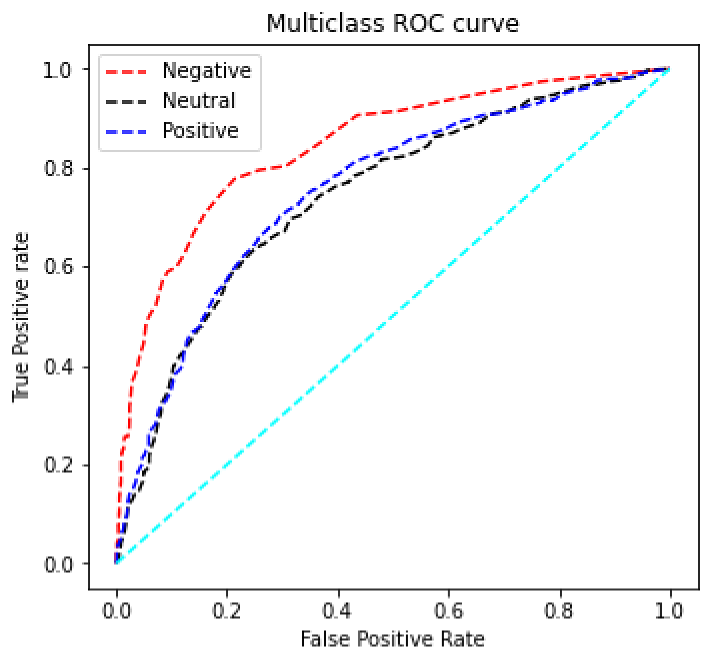
After tuning the OverSampled Logistic Regression we were able to reducing the overfitting although the model accuracy reduced from 67% (baseline logistic regrssion) to 53% on the test data set.

3. Random Forest Classifier.

In [111]:
```python
# Creating RandomOversampler Random Forest Classifier
rf_pipe = imblearn.pipeline.Pipeline([
    ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                   stop_words=stopwords_list)),
    ('os', RandomOverSampler(random_state=42)),
    ('clf', RandomForestClassifier(class_weight='balanced'))
])

rf_pipe.fit(X_train, y_train)
y_pred = rf_pipe.predict(X_test)
multiclass_eval(y_test, y_pred, X_test, X_train, rf_pipe)
```

```
Training score: 0.9956527836208106        Test Score:0.6758272574312956


Classifictaion Report
_______________________________________
              precision    recall  f1-score   support

           0       0.57      0.21      0.30       117
           1       0.71      0.82      0.76      1077
           2       0.59      0.51      0.55       589

    accuracy                           0.68      1783
   macro avg       0.63      0.51      0.54      1783
weighted avg       0.66      0.68      0.66      1783
```





The model performed better than the RandomOversampled Logistic Regrssion and the tuned one with 68% accuracy although slightly overfitting on the training dataset.

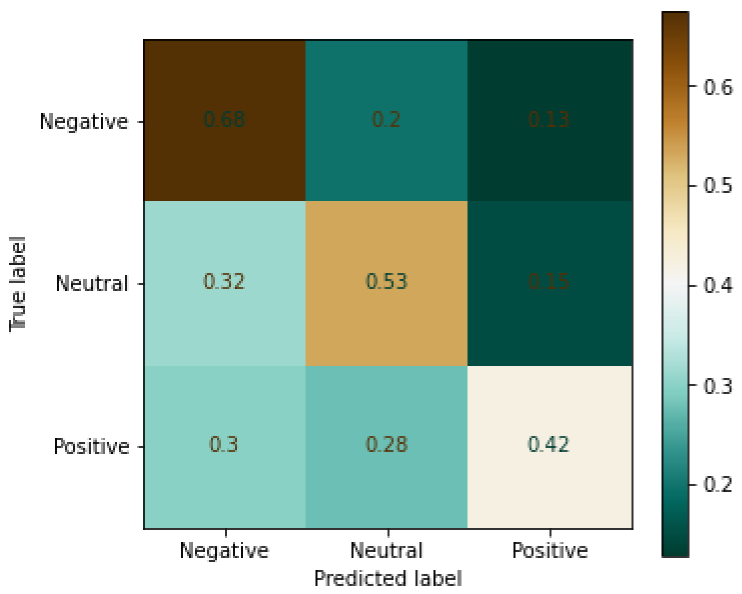
Hyperparameter Tuning with GridSearchCV

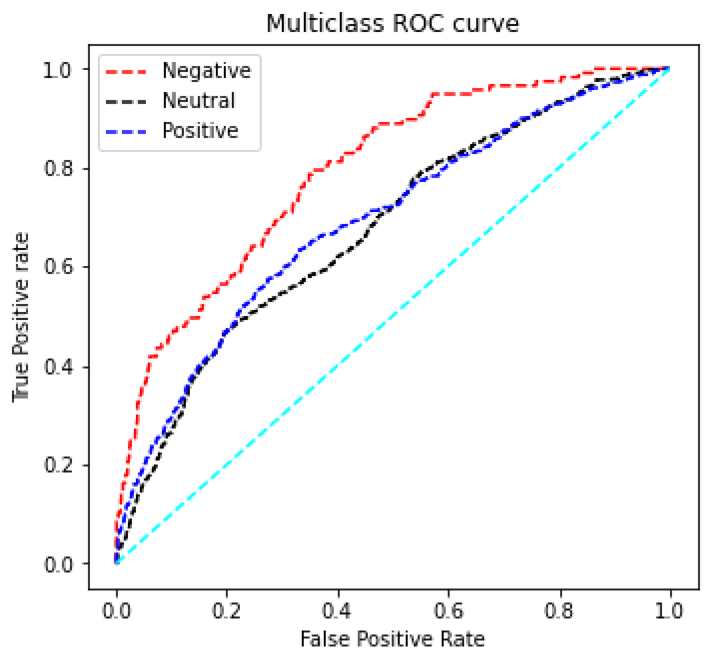
In [112]:
```python
# Performing a GridSearch to determine best parametors.
param_grid = {
    'clf__criterion': ['gini', 'entropy'],
    'clf__max_depth': [2, 5, 10],
    'clf__min_samples_split': [1, 5, 10, 20],
    'clf__min_samples_leaf': [1, 5, 10],
    'clf__n_estimators':[50, 100]
}

gs_rf = GridSearchCV(estimator=rf_pipe, param_grid=param_grid, scoring='recall_macro')
gs_rf.fit(X_train, y_train)
gs_rf.best_params_
```

Out[112]:
```
{'clf__criterion': 'gini',
 'clf__max_depth': 10,
 'clf__min_samples_leaf': 1,
 'clf__min_samples_split': 10,
 'clf__n_estimators': 100}
```

In [114]:
```
rf_pipe = imblearn.pipeline.Pipeline([
    ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                   stop_words=stopwords_list)),
    ('os', RandomOverSampler(random_state=42)),
    ('clf', RandomForestClassifier(class_weight='balanced',
                                   criterion='entropy',
                                   max_depth=10,
                                   min_samples_leaf=1,
                                   min_samples_split=10,
                                   n_estimators=100))
])
rf_pipe.fit(X_train, y_train)
y_pred = rf_pipe.predict(X_test)
multiclass_eval(y_test, y_pred, X_test, X_train, rf_pipe)
```

```
Training score: 0.5366708736502595        Test Score:0.5058889512058329


Classifictaion Report
_______________________________________
              precision    recall  f1-score   support

           0       0.13      0.68      0.22       117
           1       0.75      0.53      0.63      1077
           2       0.58      0.42      0.49       589

    accuracy                           0.51      1783
   macro avg       0.49      0.54      0.45      1783
weighted avg       0.66      0.51      0.55      1783
```



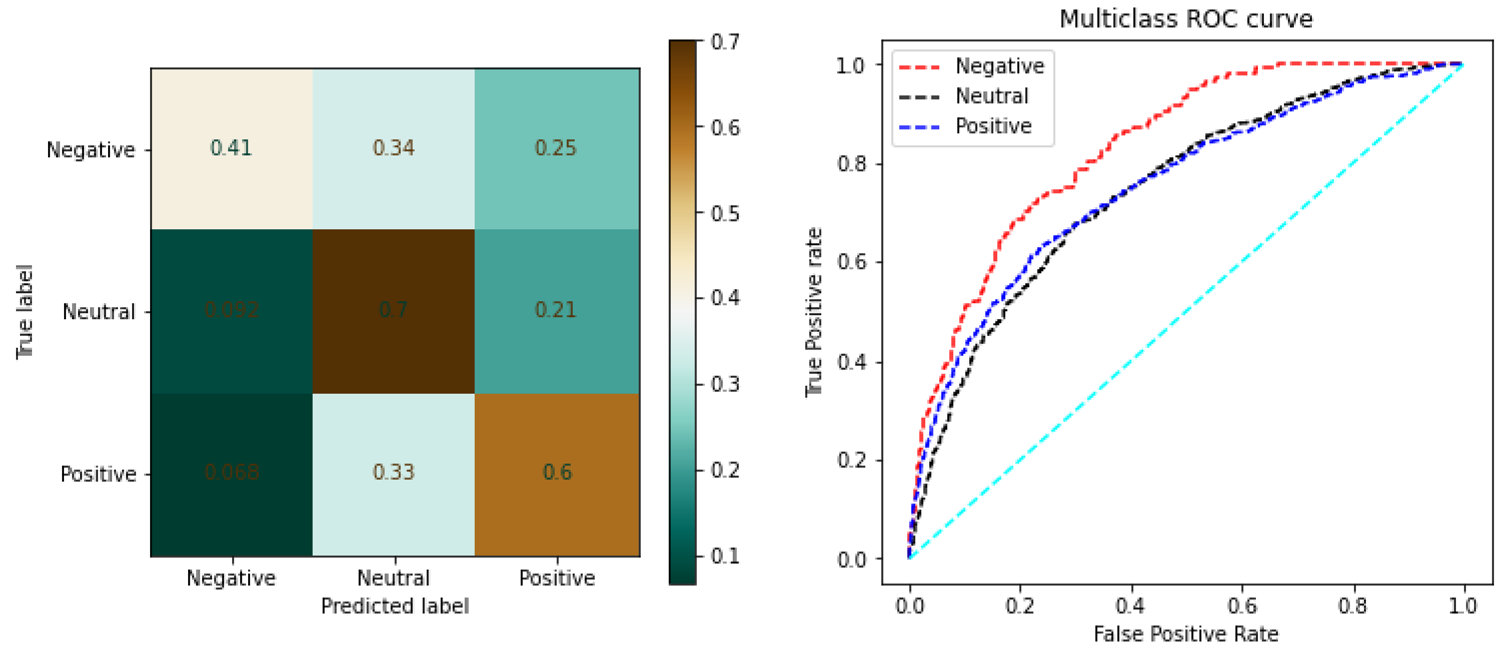


GridSearchCV tuning was able to reduce overfitting on the training data but also reduced the accuracy on the testing data.

4. XGBoost Classifier

In [117]:
```
# Creating RandomOversampler with XGboost
xgb_pipe = imblearn.pipeline.Pipeline([
    ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                   stop_words=stopwords_list)),
    ('os', RandomOverSampler(random_state=42)),
    ('xgb', XGBClassifier(random_state=42))
])

xgb_pipe.fit(X_train, y_train)
y_pred = rf_pipe.predict(X_test)
multiclass_eval(y_test, y_pred, X_test, X_train, xgb_pipe)
```

```
Training score: 0.8498106857383256        Test Score:0.6477846326416152


Classifictaion Report
_______________________________________
              precision    recall  f1-score   support

           0       0.13      0.68      0.22       117
           1       0.75      0.53      0.63      1077
           2       0.58      0.42      0.49       589

    accuracy                           0.51      1783
   macro avg       0.49      0.54      0.45      1783
weighted avg       0.66      0.51      0.55      1783
```



The **XGBoost model** was able to perform better than the logistic regrssion model. The accuracy was low about 51%. So we decided to do a GridSearchCV to get the best parametors that can help to improve the model.

Hyperparameter Tuning with GridSearchCV

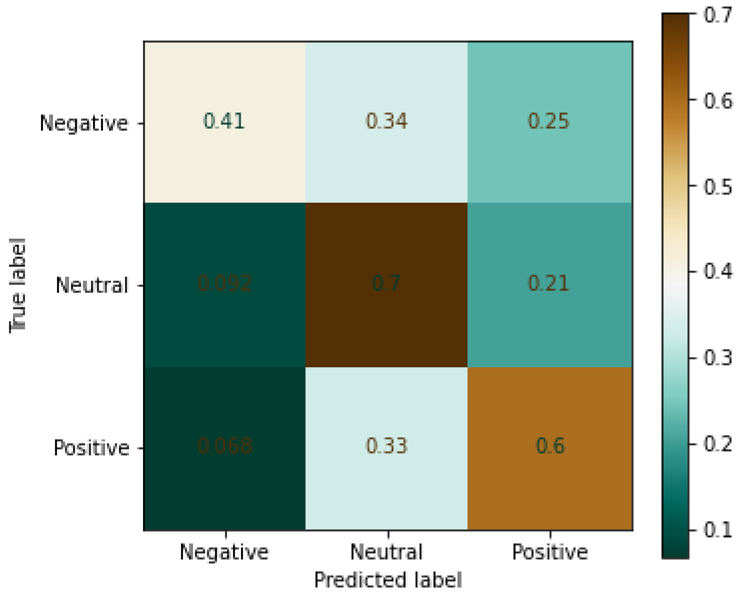
In [118]:
```
# setting basic parameter
xgb_param_grid = [{'xgb__eta': [0.1, 0.2, 0.3, 0.4],
         'xgb__gamma': [len(range(1, 50))],
         'xgb__max_depth': [len(range(1, 10))],
         'xgb__subsample': [len(range(0,1))],
         'xgb__booster': ['gbtree', 'dart']}]

# Creating a GridSearchCV with xgb_model as the estimator
gs_xgb= GridSearchCV(estimator=xgb_pipe,
                       param_grid=xgb_param_grid,
                       cv=5,
                       scoring='recall_macro',
                       verbose=1)

# Fitting the model
gs_xgb.fit(X_train, y_train)

# Get pest parametors
gs_xgb.best_params_
```

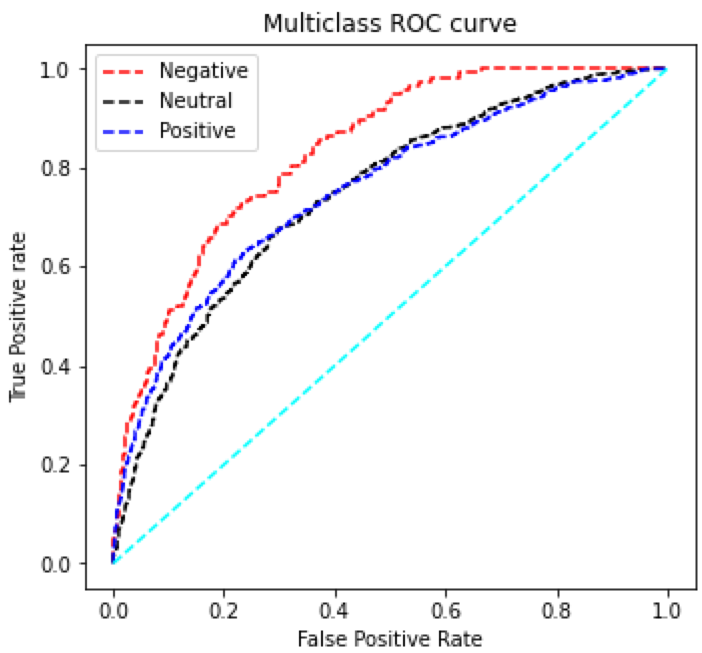
```
Fitting 5 folds for each of 8 candidates, totalling 40 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed: 15.6min finished
```

Out[118]:
```
{'xgb__booster': 'gbtree',
 'xgb__eta': 0.4,
 'xgb__gamma': 49,
 'xgb__max_depth': 9,
 'xgb__subsample': 1}
```

In [132]:
```
gs_xgb_pipe = imblearn.pipeline.Pipeline([
    ('vectorizer', TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                   stop_words=stopwords_list)),
    ('os', RandomOverSampler(random_state=42)),
    ('clf', XGBClassifier(xgb__booster='gbtree',
                          xgb__eta= 0.4,
                          xgb__gamma=49,
                          xgb__max_depth=9,
                          xgb__subsample=1
                          ))
])
model = gs_xgb_pipe.fit(X_train, y_train)
y_pred = rf_pipe.predict(X_test)
multiclass_eval(y_test, y_pred, X_test, X_train, gs_xgb_pipe)
```

```
[14:30:59] WARNING: C:\Users\dev-admin\croot2\xgboost-split_1675461376218\work\src\learner.cc:767:
Parameters: { "xgb__booster", "xgb__eta", "xgb__gamma", "xgb__max_depth", "xgb__subsample" } are not used.

Training score: 0.8498106857383256        Test Score:0.6477846326416152


Classifictaion Report
_______________________________________
              precision    recall  f1-score   support

           0       0.13      0.68      0.22       117
           1       0.75      0.53      0.63      1077
           2       0.58      0.42      0.49       589

    accuracy                           0.51      1783
   macro avg       0.49      0.54      0.45      1783
weighted avg       0.66      0.51      0.55      1783
```





After **GridSearchCV** the XGBoost model had the same score on the test dataset prediction of 0.65 and a training dataset prediction of 0.85 and the same lower accuracy score of 0.51 before and after GridSearcCV tuning.

6. Evaluation

`Evaluating the Binary models`

In the binary classification section, we focused on optimizing our models for the F1 score since we would like our model to predict both negative and positive tweets correctly. For the binary classification problem, the best model was the tuned random oversampled Multinomial Naive Bayes model based on the score of 0.61 on the training dataset, and an F1 score of 0.58.

`Multiclass models`

In the multiclass classification the best model in this task was the tuned oversampled logistic regression model with am accuracy score of 0.67. The tuned random oversampled logistic regression model also had a score of 0.53 on the testing dataset and 0.54 on the training dataset; however, since it got the highest accuracy score compared to the rest, we declaring the best model in this task.

XGBoost model had the same score on the test dataset prediction of 0.65 and a test dataset prediction of 0.85 but with a lower accuracy score of 0.51 before and after GridSearcCV tuning.

Since this is a multiclass model, it is more difficult to interpret how each word is affecting the prediction; however, it still provides insight into important words that Apple and Google should keep an eye out for which include words like: “case”, “map”, "headaches", "#fail", "hate", "battery" etc.

7. Recommendations

- Improve Customer Service: As there were complaints regarding customer service, ensuring quality service is provided would help improve the brand reputation
- From the look of analysis we did, it seem that users are not happy with iPhone's battery performance and therefore more research in needed this area to improve their products.
- Monitoring social media regularly to stay informed about public opinions and respond promptly to customer feedback
- Competitor Analysis: Extend sentiment analysis to include competitors' products and events to understand customer perceptions and identify opportunities for differentiation and improvement.

8. Conclusion

In today's world, it is imperative for businesses to attentively listen to their customers. Actively paying heed to public opinion regarding their products and services is not only crucial for maintaining financial success but also opens doors for remaining competitive in the market. By utilizing the models we have developed, Apple and Google can effectively monitor the sentiment surrounding their events and products on various social media platforms. This approach would enable the company to stay well-informed about what people are expressing regarding their competitors, potentially granting them a competitive edge.

8.1 Limitations of our analysis

The dataset utilized for our study was sourced from a crowd, which presents specific challenges.

One such challenge involves the subjective nature of labeling tweets as "Positive," "Negative," or "No emotion." What may be perceived as positive by me could be interpreted as negative by someone else.

Moreover, the contextual nuances of these tweets hold significance. Considering the absence of information regarding the labeling methodology employed, the potential for human error in labeling arises.

For instance, a tweet intended to be sarcastic may have been inaccurately labeled. This situation detrimentally affects the data's quality.

8.2 Next Steps

The companies would like to generalize these models for different applications, we would definitely gather more data from Twitter and potentially looking at other sources, such as Instagram and even generating data from TikTok.