1 Tweet Analysis - Apple and Google

Author: Joseph Denney

Email: [joseph.d.denney@gmail.com (mailto:joseph.d.denney@gmail.com)](mailto:joseph.d.denney@gmail.com)

github: [www.github.com/josephdenney/Tweet_Analysis (http://www.github.com/josephdenney/Tweet_Analysis)](http://www.github.com/josephdenney/Tweet_Analysis)

1.1 Introduction

1.1.1 Problem and Purpose

A client is looking to design and manufacture a new smart phone and will invariably compete with Apple and Google products. They have provided us with a data set of Tweets and would like more detail regarding negatively and positively charged Tweets directed at both iPhone OS and Android OS phones.

Our challenges are -

** 1. To highlight any negative features of iPhones and Androids so that they can reduce them in their new product and*

** 2. To highlight positive features of iPhones and Androids so that they can implement or improve them in their own product*

** 3. To provide recommendations that will improve their future product*

1.2 Table of Contents

1.3 EDA and Data Preprocessing

1.3.1 Library, function, and data imports

In [2]:

```
import numpy as np
import pandas as pd
import spacy
import re
import nltk
import matplotlib.pyplot as plt
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
                    level=logging.INFO)

from gensim.models import Word2Vec
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import MinMaxScaler, MaxAbsScaler
import seaborn as sns

from nltk.stem.wordnet import WordNetLemmatizer
import string
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
from sklearn.pipeline import Pipeline
from nltk.corpus import stopwords
from nltk import word_tokenize, FreqDist
from applesauce import model_scoring, cost_benefit_analysis, evaluate_model
from applesauce import model_opt, single_model_opt

from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import plot_confusion_matrix, accuracy_score
from sklearn.metrics import precision_recall_curve, f1_score, precision_score
from sklearn.metrics import recall_score
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.naive_bayes import BernoulliNB, CategoricalNB, GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.utils import resample

from keras.preprocessing.sequence import pad_sequences
from keras.layers import Input, Dense, LSTM, Embedding
from keras.layers import Dropout, Activation, Bidirectional, GlobalMaxPool1D
from keras.models import Sequential
from keras import initializers, regularizers, constraints, optimizers, layers
from keras.preprocessing import text, sequence
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\josep\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\josep\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\josep\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [3]:
```
nlp = spacy.load("en_core_web_sm")
```

In [4]:
```
print(stopwords)
print(nlp.Defaults.stop_words)
# view list of stopwords
```

```
<WordListCorpusReader in '.../corpora/stopwords' (not loaded yet)>
{'go', 'whence', 'over', 'down', 'two', 'on', 'never', 'other', 'through',
'about', '’ll', 'herself', 'are', 'whereby', 'within', 'became', 'keep',
"'re", 'full', 'fifty', '’ve', 'is', 'us', 'further', 'this', 'nor', 'tak
e', 'being', 'indeed', 'same', 'across', 'should', '‘s', 'without', 'eigh
t', 'thru', 'do', 'that', 'every', 'give', '’m', 'before', 'did', 'at', 'wo
uld', 'whereupon', 'you', 'latter', 'noone', 'ten', 'an', 'whereafter', 'an
d', 'yourselves', 'both', 'mostly', 'beside', 'beyond', 'seem', 'any', 'for
ty', 'beforehand', 'thence', 'where', 'anyhow', 'anyone', 'these', 'hers',
'upon', 'had', 'itself', 'becomes', 'part', 'there', 'was', 'quite', 'thei
r', 'amongst', 'however', 'sometimes', 'anyway', 'since', 'whom', 'almost',
'everyone', 'it', 'four', 'less', 'does', 'move', 'wherein', 'have', 'vario
us', 'by', 're', 'thereby', '‘d', 'n’t', 'enough', 'behind', 'due', 'none',
'hereupon', 'one', '’s', 'were', 'mine', 'moreover', 'least', 'among',
"'d", 'but', 'ours', 'make', 'while', 'seemed', 'really', 'ca', 'regardin
g', 'rather', 'they', 'into', 'whole', 'ourselves', 'which', 'back', 'sixt
y', "n't", 'formerly', 'alone', 'doing', 'off', 'such', 'twenty', 'above',
'those', 'than', 'themselves', 'been', "'s", 'myself', 'twelve', 'another',
'somewhere', 'a', '’re', 'unless', 'using', 'meanwhile', 'my', 'thereafte
r', 'nobody', 'how', 'what', 'serious', 'sometime', 'very', 'become', 'hund
red', 'am', 'first', 'nothing', 'yourself', 'can', 'who', '‘ll', 'third',
'most', 'either', "'ll", 'everything', 'must', 'although', '‘ve', 'see', 't
hough', 'then', 'show', 'three', 'too', 'perhaps', 'many', 'whoever', 'som
e', 'elsewhere', 'thereupon', 'once', 'our', '’d', 'becoming', 'neverthele
s', 'front', 'ever', 'put', 'much', 'still', 'used', 'namely', 'seems', 'ne
xt', 'please', "'ve", 'throughout', 'made', 'until', 'last', 'eleven', '‘r
e', 'well', 'together', 'could', 'neither', 'afterwards', 'anywhere', 'me',
'might', 'with', 'here', 'somehow', 'out', 'under', 'fifteen', 'already',
'wherever', 'else', 'thus', 'whither', 'if', "'m", 'during', 'also', 'she',
'below', 'onto', 'all', 'five', 'bottom', 'whenever', 'because', 'always',
'call', 'we', 'therein', 'be', 'besides', 'between', 'after', 'per', 'via',
'himself', 'no', 'more', 'amount', 'six', 'latterly', 'the', 'everywhere',
'yours', 'something', 'often', 'only', 'each', 'side', 'others', 'i', 'here
after', 'them', 'herein', 'nowhere', 'cannot', 'he', 'hence', 'in', 'agai
n', 'anything', 'his', 'nine', 'whether', 'few', 'to', 'get', 'so', 'even',
'toward', 'along', 'except', 'empty', 'its', 'not', 'former', 'why', 'you
r', 'several', 'otherwise', 'yet', 'around', 'therefore', 'of', 'from', 'he
r', 'now', 'done', 'top', 'whereas', 'him', '‘m', 'against', 'for', 'up',
'when', 'whose', 'just', 'someone', 'has', 'towards', 'n‘t', 'say', 'may',
'as', 'name', 'or', 'own', 'hereby', 'whatever', 'seeming', 'will'}
```

In [5]:
```
df = pd.read_csv('data/product_tweets.csv',encoding='latin1')
```

In [6]:
```
df.info()
```

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

In [7]:
```
df.head()
```

Out[7]:

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_or_
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	iPad or iPhone App	Negative

In [8]:
```
df['emotion_in_tweet_is_directed_at'].unique()
```

Out[8]:
```
array(['iPhone', 'iPad or iPhone App', 'iPad', 'Google', nan, 'Android',
       'Apple', 'Android App', 'Other Google product or service',
       'Other Apple product or service'], dtype=object)
```

In [9]:
```
df['emotion_in_tweet_is_directed_at'].count()
```

Out[9]:
```
3291
```

1.3.2 Data Exploration and Column Title Cleanup

In [10]:
```
df['is_there_an_emotion_directed_at_a_brand_or_product'].unique()
```

Out[10]:
```
array(['Negative emotion', 'Positive emotion',
       'No emotion toward brand or product', "I can't tell"], dtype=object)
```

In [11]:
```
df = df.rename(columns= {'is_there_an_emotion_directed_at_a_brand_or_product'
                        :'Emotion',
                        'emotion_in_tweet_is_directed_at': 'Platform'})
```

In [12]:
```
df = df.rename(columns= {'tweet_text': 'Tweet'})
```

In [13]:
```
df.head()
```

Out[13]:

	Tweet	Platform	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

In [14]:
```
df.groupby(df['Platform']).count()
```

Out[14]:

	Tweet	Emotion
Platform		
Android	78	78
Android App	81	81
Apple	661	661
Google	430	430
Other Apple product or service	35	35
Other Google product or service	293	293
iPad	946	946
iPad or iPhone App	470	470
iPhone	297	297

1.3.3 Dummify Target Column

In [15]:
```
df_dummify = pd.get_dummies(df['Emotion'])
```

In [16]:
```
df_dummify.head()
```

Out[16]:

	I can't tell	Negative emotion	No emotion toward brand or product	Positive emotion
0	0	1	0	0
1	0	0	0	1
2	0	0	0	1
3	0	1	0	0
4	0	0	0	1

In [17]:
```
df_dummify.sum() # class bias
```

Out[17]:
```
I can't tell                            156
Negative emotion                        570
No emotion toward brand or product     5389
Positive emotion                       2978
dtype: int64
```

In [18]:
```
df.info()
df = pd.merge(df, df_dummify, how='outer',on=df.index)
# ran this code, dummify emotion data
```

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

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9093 entries, 0 to 9092
Data columns (total 8 columns):
 #   Column                              Non-Null Count  Dtype 
---  ------                              --------------  ----- 
 0   key_0                               9093 non-null   int64 
 1   Tweet                               9092 non-null   object
 2   Platform                            3291 non-null   object
 3   Emotion                             9093 non-null   object
 4   I can't tell                        9093 non-null   uint8 
 5   Negative emotion                    9093 non-null   uint8 
 6   No emotion toward brand or product  9093 non-null   uint8 
 7   Positive emotion                    9093 non-null   uint8 
dtypes: int64(1), object(3), uint8(4)
memory usage: 390.7+ KB
```

In [20]:

```
df.head()
```

Out[20]:

	key_0	Tweet	Platform	Emotion	I can't tell	Negative emotion	No emotion toward brand or product	Positive emotion
0	0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative emotion	0	1	0	0
1	1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive emotion	0	0	0	1
2	2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive emotion	0	0	0	1
3	3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Negative emotion	0	1	0	0
4	4	@sxtxstate great stuff on Fri #SXSW:	Google	Positive emotion	0	0	0	1

In [21]:

```
df = df.rename(columns = {"I can't tell": "Uncertain",
                          'Negative emotion': 'Negative',
                          'No emotion toward brand or product': 'No Emotion',
                          'Positive emotion':'Positive'})
```

In [22]:

```
df = df.drop(columns='key_0')
df.head()
df.to_csv('Full_DF')
```

In [23]:
```
corpus = list(df['Tweet']) # verify corpus list
corpus[:10]
```

Out[23]:
```
['.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it w
as dead!  I need to upgrade. Plugin stations at #SXSW.',
 "@jessedee 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 down at
#SXSW.',
 "@sxsw I hope this year's festival isn't as crashy as this year's iPhone a
pp. #sxsw",
 "@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly
(tech books/conferences) &amp; Matt Mullenweg (Wordpress)",
 '@teachntech00 New iPad Apps For #SpeechTherapy And Communication Are Show
cased At The #SXSW Conference http://ht.ly/49n4M (http://ht.ly/49n4M) #iear
#edchat #asd',
 nan,
 '#SXSW is just starting, #CTIA is around the corner and #googleio is only
 a hop skip and a jump from there, good time to be an #android fan',
 'Beautifully smart and simple idea RT @madebymany @thenextweb wrote about
 our #hollergram iPad app for #sxsw! http://bit.ly/ieaVOB', (http://bit.ly/
ieaVOB',)
 'Counting down the days to #sxsw plus strong Canadian dollar means stock u
p on Apple gear']
```

1.3.4 Platform Negative Tweet Table

In [24]:
```
df.groupby(by=df['Platform']).sum()
```

Out[24]:

Platform	Uncertain	Negative	No Emotion	Positive
Android	0.0	8.0	1.0	69.0
Android App	0.0	8.0	1.0	72.0
Apple	2.0	95.0	21.0	543.0
Google	1.0	68.0	15.0	346.0
Other Apple product or service	0.0	2.0	1.0	32.0
Other Google product or service	1.0	47.0	9.0	236.0
iPad	4.0	125.0	24.0	793.0
iPad or iPhone App	0.0	63.0	10.0	397.0
iPhone	1.0	103.0	9.0	184.0

1.3.5 Tokenize and Create Bag of Words

In [25]:
```
tokenz = word_tokenize(','.join(str(v) for v in corpus))
```

In [26]:
```python
tokenz[:10]
```
Out[26]:
```
['.', '@', 'wesley83', 'I', 'have', 'a', '3G', 'iPhone', '.', 'After']
```

1.3.6 Create Stopwords List

In [27]:
```python
stopword_list = list(nlp.Defaults.stop_words)
len(nlp.Defaults.stop_words)
```
Out[27]:
```
326
```

In [28]:
```python
stopword_list
```
Out[28]:
```
['go',
 'whence',
 'over',
 'down',
 'two',
 'on',
 'never',
 'other',
 'through',
 'about',
 '’ll',
 'herself',
 'are',
 'whereby',
 'within',
 'became',
 'keep',
 "'re",
 'full',
```

In [29]:
```python
stopword_list.extend(string.punctuation)
```

In [30]:
```python
len(stopword_list)
```
Out[30]:
```
358
```

In [31]:
```python
stopword_list.extend(stopwords.words('english'))
```

In [32]:
```python
len(stopword_list)
```
Out[32]:
```
537
```

In [33]:
```python
additional_punc = ['“','”','...',"''",'’','``','https','rt','\.+']
stopword_list.extend(additional_punc)
stopword_list[-10:]
```
Out[33]:
```
["wouldn't", '“', '”', '...', "''", '’', '``', 'https', 'rt', '\\.+']
```

1.3.7 Remove Stopwords and Additional Punctuation from the

Data

In [34]:
```
stopped_tokenz = [word.lower() for word in tokenz if word.lower()
                  not in stopword_list]
```

In [35]:
```
freq = FreqDist(stopped_tokenz)
freq.most_common(50)
```

Out[35]:
```
[('sxsw', 9418),
 ('mention', 7120),
 ('link', 4313),
 ('google', 2593),
 ('ipad', 2432),
 ('apple', 2301),
 ('quot', 1696),
 ('iphone', 1516),
 ('store', 1472),
 ('2', 1114),
 ('new', 1090),
 ('austin', 959),
 ('amp', 836),
 ('app', 810),
 ('circles', 658),
 ('launch', 653),
 ('social', 647),
 ('android', 574),
 ('today', 574),
 ('network', 465),
 ('ipad2', 457),
 ('pop-up', 420),
 ('line', 405),
 ('free', 387),
 ('called', 361),
 ('party', 346),
 ('sxswi', 340),
 ('mobile', 338),
 ('major', 301),
 ('like', 290),
 ('time', 271),
 ('temporary', 264),
 ('opening', 257),
 ('possibly', 240),
 ('people', 226),
 ('downtown', 225),
 ('apps', 224),
 ('great', 222),
 ('maps', 219),
 ('going', 217),
 ('check', 216),
 ('mayer', 214),
 ('day', 214),
 ('open', 210),
 ('popup', 209),
 ('need', 205),
 ('marissa', 189),
 ('got', 185),
 ('w/', 182),
 ('know', 180)]
```

1.3.8 Lemmatize the Data, Utilize Regex to Find and Remove URL's, Tags, other Misc

In [36]:
```
additional_misc = ['sxsw','mention',r'[a-zA-Z]+\'?s]',
                   r"(http[s]?://\w*\.\w*/+\w+)", r'\#\w*',
                   r'RT [@]?\w*:', r'\@\w*',r"\d$",r"^\d",
                   r"([a-zA-Z]+(?:'[a-z]+)?)",r'\d.',r'\d','RT',
                   r'^http[s]?','za'] #[A-Z]{2,20} remove caps like MAGA and
stopword_list.extend(additional_misc)
stopword_list.extend(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'])
```

In [37]:
```
lemmatizer = WordNetLemmatizer()
```

In [38]:
```
clean_stopped_tokenz = [word.lower() for word in stopped_tokenz if word
                        not in stopword_list]
clean_lemmatized_tokenz = [lemmatizer.lemmatize(word.lower()) for word
                           in stopped_tokenz if word not in stopword_list]
```

In [39]:
```
freq_clean_lemma = FreqDist(clean_lemmatized_tokenz)
freq_lemma = freq_clean_lemma.most_common(5000)
freq_lemma2 = freq_clean_lemma.most_common(25)
```

In [40]:
```
total_word_count = len(clean_lemmatized_tokenz)
```

In [41]:
```
lemma_word_count = sum(freq_clean_lemma.values()) # just a number
```

In [42]:
```
for word in freq_lemma2: # separate both classes, positive and negative
    normalized_freq = word[1] / lemma_word_count
    print(word, "----", "{:.3f}".format(normalized_freq*100),"%")
```

```
('link', 4324) ---- 5.004 %
('google', 2594) ---- 3.002 %
('ipad', 2432) ---- 2.814 %
('apple', 2304) ---- 2.666 %
('quot', 1696) ---- 1.963 %
('iphone', 1516) ---- 1.754 %
('store', 1511) ---- 1.749 %
('new', 1090) ---- 1.261 %
('austin', 960) ---- 1.111 %
('amp', 836) ---- 0.967 %
('app', 810) ---- 0.937 %
('launch', 691) ---- 0.800 %
('circle', 673) ---- 0.779 %
('social', 647) ---- 0.749 %
('android', 574) ---- 0.664 %
('today', 574) ---- 0.664 %
('network', 473) ---- 0.547 %
('ipad2', 457) ---- 0.529 %
('line', 442) ---- 0.512 %
('pop-up', 422) ---- 0.488 %
('free', 387) ---- 0.448 %
('party', 386) ---- 0.447 %
('called', 361) ---- 0.418 %
('mobile', 340) ---- 0.393 %
('sxswi', 340) ---- 0.393 %
```

In [43]:
```
# from wordcloud import WordCloud

# ## Initalize a WordCloud with our stopwords_list and no bigrams
# wordcloud = WordCloud(stopwords=stopword_list,collocations=False)

# ## Generate wordcloud from stopped_tokens
# wordcloud.generate(','.join(clean_lemmatized_tokenz))

# ## Plot with matplotlib
# plt.figure(figsize = (12, 12), facecolor = None)
# plt.imshow(wordcloud)
# plt.axis('off')
```

In [44]:
```
bigram_measures = nltk.collocations.BigramAssocMeasures()
tweet_finder = nltk.BigramCollocationFinder.from_words(clean_lemmatized_toker
tweets_scored = tweet_finder.score_ngrams(bigram_measures.raw_freq)
```

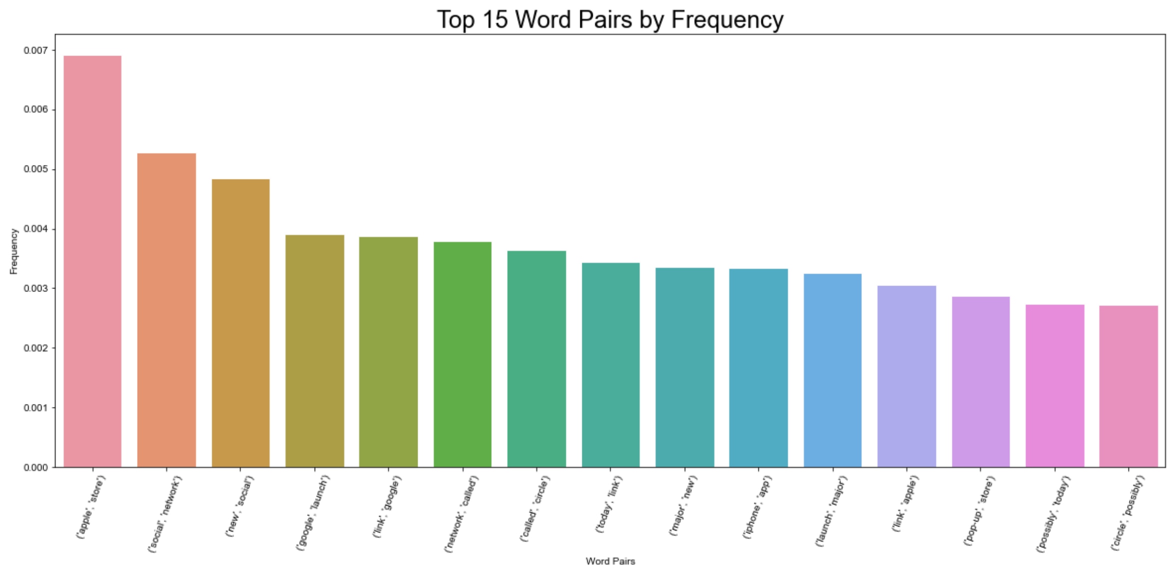
In [45]:
```
word_pairs = pd.DataFrame(tweets_scored, columns=["Word","Freq"]).head(20)
word_pairs
```

Out[45]:

	Word	Freq
0	(apple, store)	0.006920
1	(social, network)	0.005277
2	(new, social)	0.004837
3	(google, launch)	0.003912
4	(link, google)	0.003877
5	(network, called)	0.003784
6	(called, circle)	0.003634
7	(today, link)	0.003437
8	(major, new)	0.003356
9	(iphone, app)	0.003333
10	(launch, major)	0.003264

In [46]:

```
fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=word_pairs.head(15)['Word'], y=word_pairs.head(15)['Freq']
                 palette=palette)
ax.set(xlabel="Word Pairs",ylabel="Frequency")
plt.ticklabel_format(style='plain',axis='y')
plt.xticks(rotation=70)
plt.title('Top 15 Word Pairs by Frequency')
plt.show()
```



In [47]:

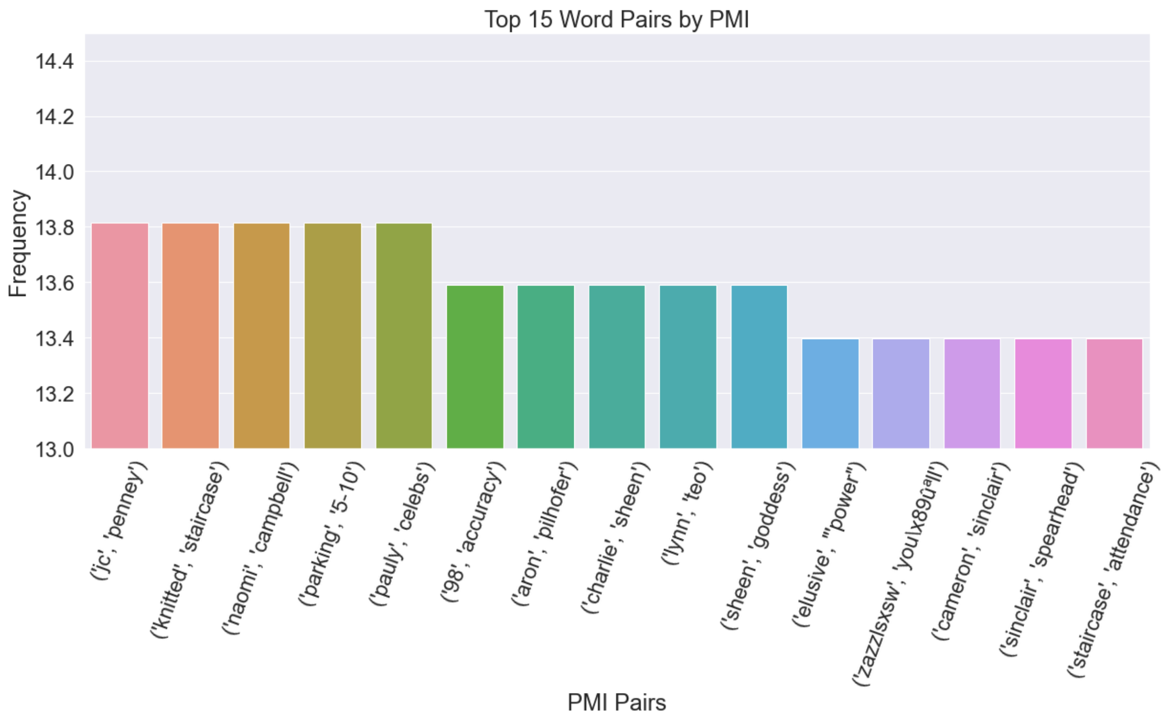
```
tweet_pmi_finder = nltk.BigramCollocationFinder.from_words(clean_lemmatized_t
tweet_pmi_finder.apply_freq_filter(5)

tweet_pmi_scored = tweet_pmi_finder.score_ngrams(bigram_measures.pmi)
```

In [48]:
```
PMI_list = pd.DataFrame(tweet_pmi_scored, columns=["Words","PMI"]).head(20)
PMI_list = PMI_list[PMI_list.PMI < 14]
PMI_list
```

Out[48]:

	Words	PMI
1	(jc, penney)	13.813948
2	(knitted, staircase)	13.813948
3	(naomi, campbell)	13.813948
4	(parking, 5-10)	13.813948
5	(pauly, celebs)	13.813948
6	(98, accuracy)	13.591556
7	(aron, pilhofer)	13.591556
8	(charlie, sheen)	13.591556
9	(lynn, teo)	13.591556
10	(sheen, goddess)	13.591556
11	(elusive, 'power)	13.398911

In [49]:
```
fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=PMI_list.head(15)['Words'], y=PMI_list.head(15)['PMI'],
                 palette=palette)
ax.set(xlabel="PMI Pairs",ylabel="Frequency")
plt.ylim([13,14.5])
plt.ticklabel_format(style='plain',axis='y')
plt.xticks(rotation=70)
plt.title('Top 15 Word Pairs by PMI')
plt.show()
```



In [50]:
```
df1 = df
df.head()
```

Out[50]:

	Tweet	Platform	Emotion	Uncertain	Negative	No Emotion	Positive
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative emotion	0	1	0	0
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive emotion	0	0	0	1
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive emotion	0	0	0	1
3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Negative emotion	0	1	0	0
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Positive emotion	0	0	0	1

In [51]:
```
df1 = df1.drop(columns=['Uncertain','No Emotion'])
# Turn negative and positive columns into one column of just negatives
# and positive.
df1 = df1[df1['Emotion'] != "No emotion toward brand or product"]
df1 = df1[df1['Emotion'] != "I can't tell"]
df1 = df1.drop(columns='Negative')
df1 = df1.rename(columns={'Positive': 'Positive_Bin'})
df1.head()
```

Out[51]:

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

In [52]:
```
df1.to_csv('Tweet.csv')
```

1.3.9 Create Upsampled Data

In [53]:
```
df_majority = df1.loc[df1['Positive_Bin']==1]
df_minority = df1.loc[df1['Positive_Bin']==0]
```

In [54]:
```
df_minority.shape
```

Out[54]: (570, 4)

In [55]:
```
df_majority.shape
```

Out[55]: (2978, 4)

In [56]:
```
df_min_sample = resample(df_minority, replace=True, n_samples=1000,
                         random_state=42)
```

In [57]:
```
df_maj_sample = resample(df_majority, replace=True, n_samples=2500,
                         random_state=42)
```

In [58]:
```
df_upsampled = pd.concat([df_min_sample, df_maj_sample], axis=0)
df_upsampled.shape
```

Out[58]: (3500, 4)

In [59]:
```
X, y = df_upsampled['Tweet'], df_upsampled['Positive_Bin']
```

In [60]:
```
df_upsampled.to_csv('Upsampled.csv')
```

1.4 Modeling

1.4.1 Train/Test Split

In [61]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [62]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3548 entries, 0 to 9088
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Tweet         3548 non-null   object
 1   Platform      3191 non-null   object
 2   Emotion       3548 non-null   object
 3   Positive_Bin  3548 non-null   uint8 
dtypes: object(3), uint8(1)
memory usage: 114.3+ KB
```

In [63]:
```
y_train.value_counts(0)
y_test.value_counts(1)
```

```
2020-12-26 12:18:47,128 : INFO : NumExpr defaulting to 8 threads.
```

Out[63]:
```
1    0.683429
0    0.316571
Name: Positive_Bin, dtype: float64
```

1.4.2 Vectorize, Lemmatize with Count Vectorizer and Tf Idf

In [64]:
```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer,
from sklearn.ensemble import RandomForestClassifier

tokenizer = nltk.TweetTokenizer(preserve_case=False)

vectorizer = CountVectorizer(tokenizer=tokenizer.tokenize,
                             stop_words=stopword_list,decode_error='ignore')
```

In [65]:
```
X_train_count = vectorizer.fit_transform(X_train)
X_test_count = vectorizer.transform(X_test)
```

```
C:\Users\josep\anaconda3\lib\site-packages\sklearn\feature_extraction\text.
py:383: UserWarning: Your stop_words may be inconsistent with your preproce
ssing. Tokenizing the stop words generated tokens [":'[", ':/', 'a-z', 'a-z
a-z', 'http', 'n', 'w', '‘'] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
```

1.4.3 MaxAbsScaler

In [66]:
```
scaler_object = MaxAbsScaler().fit(X_train_count)
```

In [67]:
```
scaler_object.transform(X_train_count)
```

Out[67]:
```
<2625x4295 sparse matrix of type '<class 'numpy.float64'>'
	with 28229 stored elements in Compressed Sparse Row format>
```

In [68]:
```
scaler_object.transform(X_test_count)
```

Out[68]:
```
<875x4295 sparse matrix of type '<class 'numpy.float64'>'
	with 8854 stored elements in Compressed Sparse Row format>
```

1.4.4 Instantiate Model

In [69]:
```
ran_for = RandomForestClassifier(class_weight='balanced')
model = ran_for.fit(X_train_count, y_train)
```

In [70]:
```
y_hat_test = model.predict(X_test_count)
```

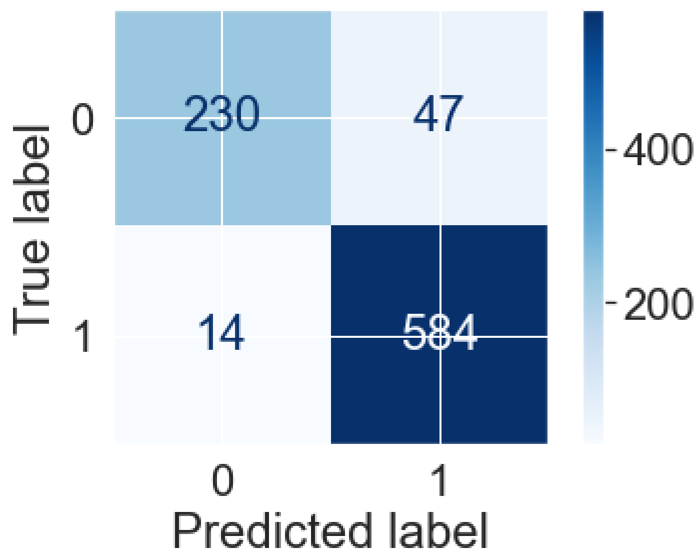
1.5 Evaluate Models

1 denotes a Positive Tweet, 0 denotes a Negative Tweet

1.5.1 Random Forest with Count Vectorizer

In [71]:
```
evaluate_model(y_test, y_hat_test, X_test_count,clf=model)
# 1 denotes Positive Tweet
```

```
              precision    recall  f1-score   support

           0       0.94      0.83      0.88       277
           1       0.93      0.98      0.95       598

    accuracy                           0.93       875
   macro avg       0.93      0.90      0.92       875
weighted avg       0.93      0.93      0.93       875
```



Basic Random Forest model performs well after preprocessing with high precision and f1-scores.

In [72]:
```
tf_idf_vectorizer = TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                    stop_words=stopword_list,
                                    decode_error='ignore')
```

In [73]:
```
X_train_tf_idf = tf_idf_vectorizer.fit_transform(X_train)
X_test_tf_idf = tf_idf_vectorizer.transform(X_test)
print(X_train_tf_idf.shape)
print(y_train.shape)
```

```
C:\Users\josep\anaconda3\lib\site-packages\sklearn\feature_extraction\text.
py:383: UserWarning: Your stop_words may be inconsistent with your preproce
ssing. Tokenizing the stop words generated tokens [":'[", ':/', 'a-z', 'a-z
a-z', 'http', 'n', 'w', '‘'] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '

(2625, 4295)
(2625,)
```

In [74]:
```
from sklearn.ensemble import RandomForestClassifier
```

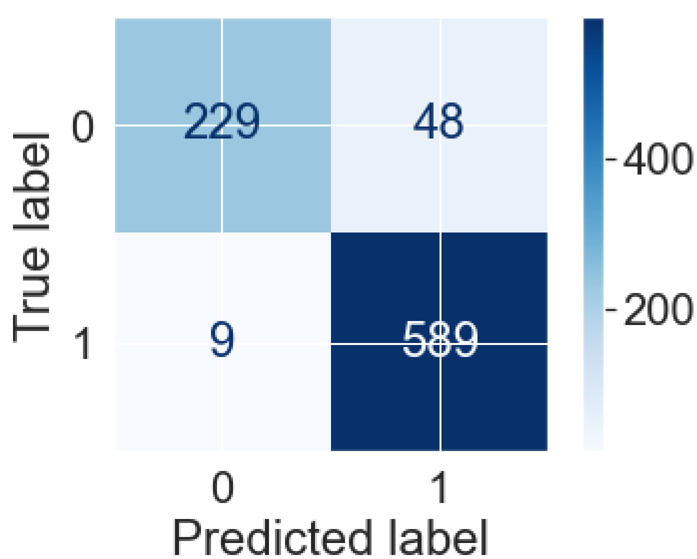
In [75]:
```
ran_for = RandomForestClassifier(class_weight='balanced')
model_tf_idf = ran_for.fit(X_train_tf_idf,y_train)
```

In [76]:
```
y_hat_tf_idf = model_tf_idf.predict(X_test_count)
```

1.5.2 Random Forest with Tf-Idf Vectorizer

In [77]:
```
evaluate_model(y_test, y_hat_tf_idf, X_test_tf_idf,clf=model_tf_idf)
```

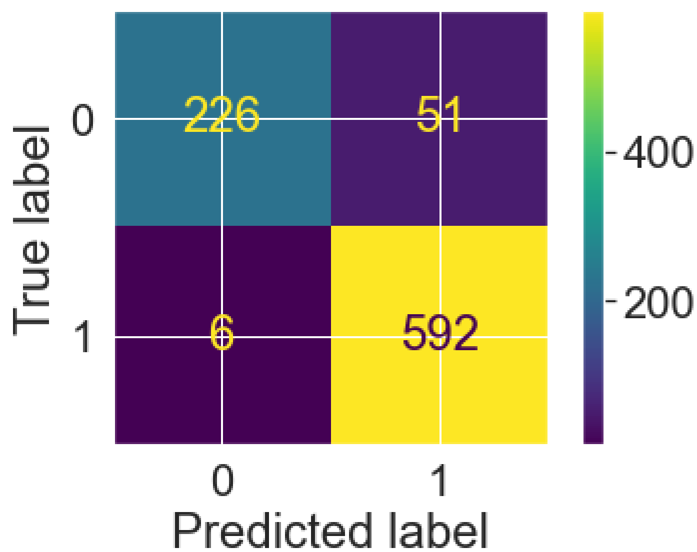
```
              precision    recall  f1-score   support

           0       0.89      0.65      0.75       277
           1       0.86      0.96      0.91       598

    accuracy                           0.86       875
   macro avg       0.87      0.81      0.83       875
weighted avg       0.87      0.86      0.86       875
```



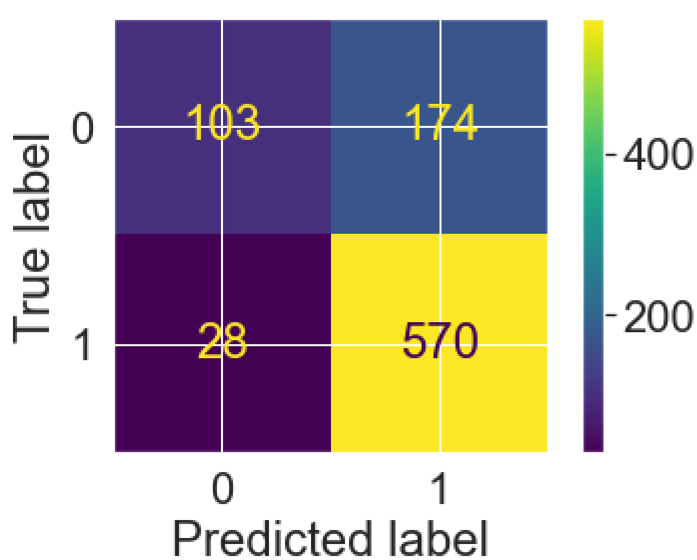
1.5.3 Multiple Models, CountVectorizer

In [78]:
```
ran_for = RandomForestClassifier()
ada_clf = AdaBoostClassifier()
gb_clf = GradientBoostingClassifier()

models = [ran_for, ada_clf, gb_clf]

for model in models:
    single_model_opt(model, X_train_count, y_train, X_test_count, y_test)
```

```
Accuracy Score:  0.9348571428571428
Precision Score:  0.9206842923794712
Recall Score:  0.9899665551839465
F1 Score:  0.9540692989524577
RandomForestClassifier()   0.9348571428571428
```

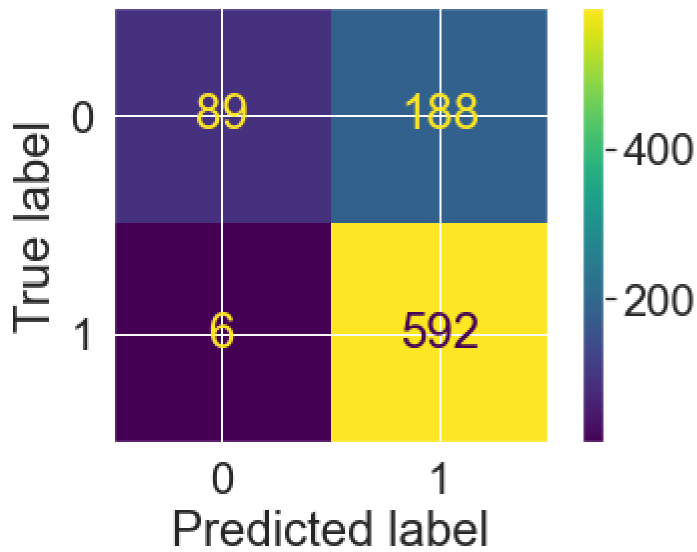



```
Accuracy Score:  0.7691428571428571
Precision Score:  0.7661290322580645
Recall Score:  0.9531772575250836
F1 Score:  0.849478390461997
AdaBoostClassifier()   0.7691428571428571
```




```
Accuracy Score:  0.7782857142857142
```

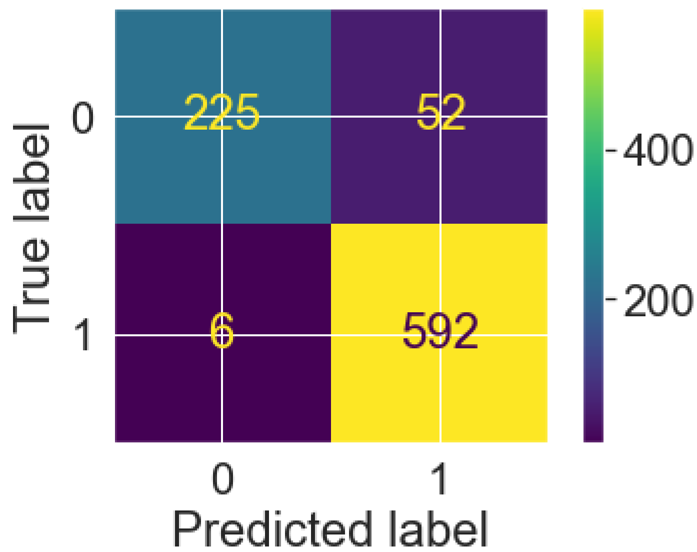
```
Precision Score:  0.7589743589743589
Recall Score:  0.9899665551839465
F1 Score:  0.8592162554426704
GradientBoostingClassifier()   0.7782857142857142
```



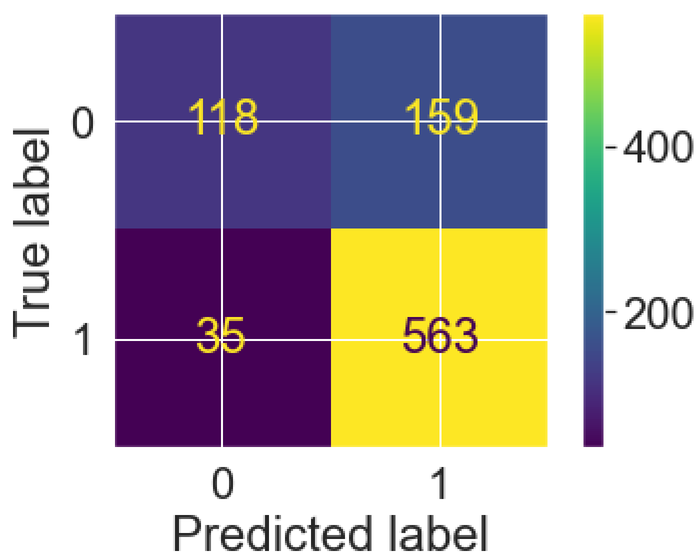
1.5.4 Multiple Models, Tf-Idf Vectorizer

In [79]:
```
for model in models:
    single_model_opt(model, X_train_tf_idf, y_train, X_test_tf_idf, y_test)
```

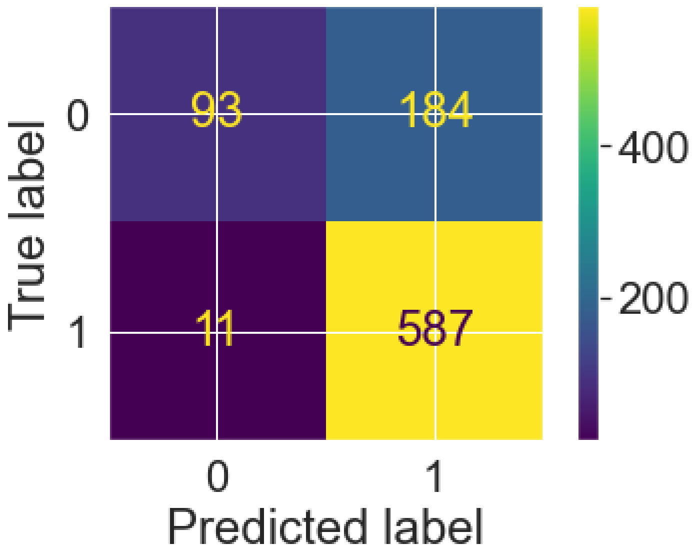
```
Accuracy Score:  0.9337142857142857
Precision Score:  0.9192546583850931
Recall Score:  0.9899665551839465
F1 Score:  0.9533011272141707
RandomForestClassifier()   0.9337142857142857
```




```
Accuracy Score:  0.7782857142857142
Precision Score:  0.7797783933518005
Recall Score:  0.9414715719063546
F1 Score:  0.8530303030303031
AdaBoostClassifier()   0.7782857142857142
```




```
Accuracy Score:  0.7771428571428571
Precision Score:  0.7613488975356679
Recall Score:  0.9816053511705686
F1 Score:  0.8575602629656682
GradientBoostingClassifier()   0.7771428571428571
```

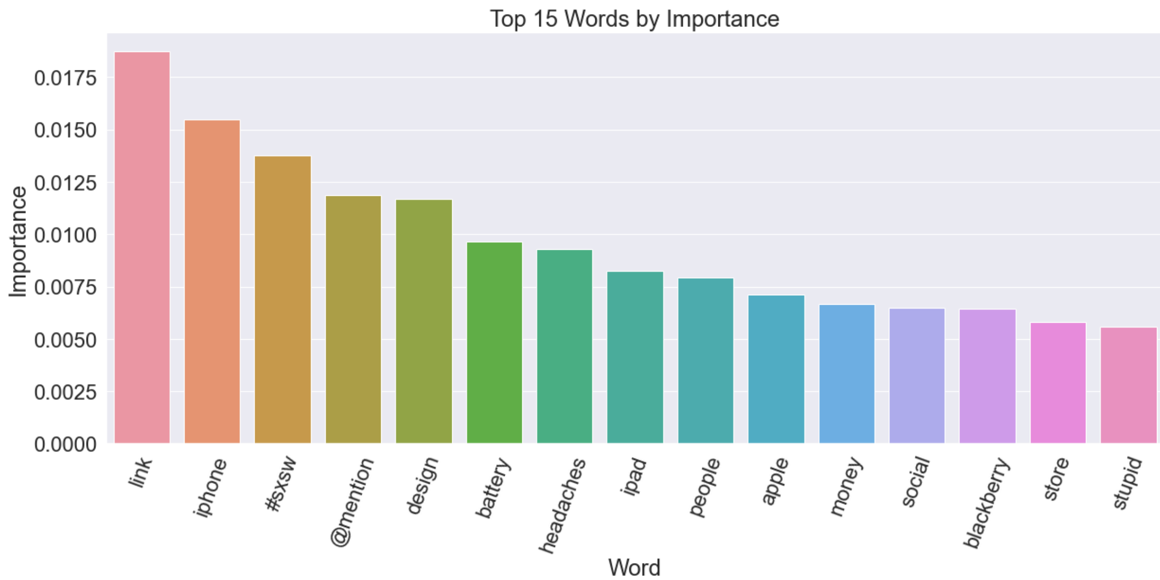


In [80]:
```
tf_idf_vectorizer.get_feature_names()
```

Out[80]:
```
['##sxsw',
 '#10',
 '#106',
 '#11ntc',
 '#1406-08',
 '#15slides',
 '#310409h2011',
 '#4sq',
 '#911tweets',
 '#abacus',
 '#accesssxsw',
 '#accordion',
 '#aclu',
 '#adam',
 '#addictedtotheinterwebs',
 '#adpeopleproblems',
 '#agchat',
 '#agileagency',
 '#agnerd',
```

In [81]:
```
importance = pd.Series(ran_for.feature_importances_,
                       index=tf_idf_vectorizer.get_feature_names())
importance = pd.DataFrame(importance).sort_values(by=0,ascending=False)
```

In [82]:
```
fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=importance.head(15).index, y=importance.head(15)[0],
                 palette=palette)
ax.set(xlabel="Word",ylabel="Importance")
plt.ticklabel_format(style='plain',axis='y')
plt.xticks(rotation=70)
plt.title('Top 15 Words by Importance')
plt.show()
```



1.5.5 Pipeline and GridSearchCV

In [83]:
```
vectorizer = CountVectorizer()
tf_transform = TfidfTransformer(use_idf=True)
```

In [84]:
```
text_pipe = Pipeline(steps=[
    ('count_vectorizer',vectorizer),
    ('tf_transformer',tf_transform)])
```

In [85]:
```
RandomForestClassifier(class_weight='balanced')
```

Out[85]:
```
RandomForestClassifier(class_weight='balanced')
```

In [86]:
```
full_pipe = Pipeline(steps=[
    ('text_pipe',text_pipe),
    ('clf',RandomForestClassifier(class_weight='balanced'))
])
```

In [87]:
```
X_train_pipe = text_pipe.fit_transform(X_train)
```

In [88]:
```
X_test_pipe = text_pipe.transform(X_test)
```

In [89]:
```
X_train_pipe
```

Out[89]:
```
<2625x4256 sparse matrix of type '<class 'numpy.float64'>'
	with 44273 stored elements in Compressed Sparse Row format>
```

In [90]:
```
params = {'text_pipe__tf_transformer__use_idf':[True, False],
         'text_pipe__count_vectorizer__tokenizer':[None,tokenizer.tokenize],
         'text_pipe__count_vectorizer__stop_words':[None,stopword_list],
         'clf__criterion':['gini', 'entropy']}
```

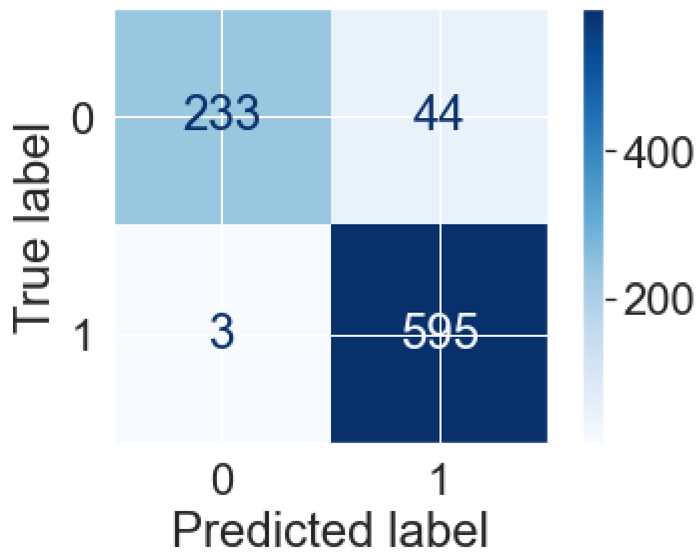
In [91]:
```
## Make and fit grid
grid = GridSearchCV(full_pipe,params,cv=3)
grid.fit(X_train,y_train)
## Display best params
grid.best_params_
```

```
C:\Users\josep\anaconda3\lib\site-packages\sklearn\feature_extraction\tex
t.py:383: UserWarning: Your stop_words may be inconsistent with your prep
rocessing. Tokenizing the stop words generated tokens ['http'] not in sto
p_words.
  warnings.warn('Your stop_words may be inconsistent with '
C:\Users\josep\anaconda3\lib\site-packages\sklearn\feature_extraction\tex
t.py:383: UserWarning: Your stop_words may be inconsistent with your prep
rocessing. Tokenizing the stop words generated tokens ['http'] not in sto
p_words.
  warnings.warn('Your stop_words may be inconsistent with '
C:\Users\josep\anaconda3\lib\site-packages\sklearn\feature_extraction\tex
t.py:383: UserWarning: Your stop_words may be inconsistent with your prep
rocessing. Tokenizing the stop words generated tokens ['http'] not in sto
p_words.
  warnings.warn('Your stop_words may be inconsistent with '
C:\Users\josep\anaconda3\lib\site-packages\sklearn\feature_extraction\tex
t.py:383: UserWarning: Your stop_words may be inconsistent with your prep
rocessing. Tokenizing the stop words generated tokens ['http'] not in sto
p_words.
```

In [92]:
```
best_pipe = grid.best_estimator_
y_hat_test = grid.predict(X_test)
```

In [93]:
```
evaluate_model(y_test,y_hat_test,X_test,best_pipe)
```

```
              precision    recall  f1-score   support

           0       0.99      0.84      0.91       277
           1       0.93      0.99      0.96       598

    accuracy                           0.95       875
   macro avg       0.96      0.92      0.94       875
weighted avg       0.95      0.95      0.95       875
```



In [94]:
```
X_train_pipe.shape
```

Out[94]: (2625, 4256)

1.5.6 Bigram Frequency

In [95]:
```
features = text_pipe.named_steps['count_vectorizer'].get_feature_names()
features[:10]
```

Out[95]: ['000', '02', '03', '0310apple', '08', '10', '100', '100s', '101', '106']

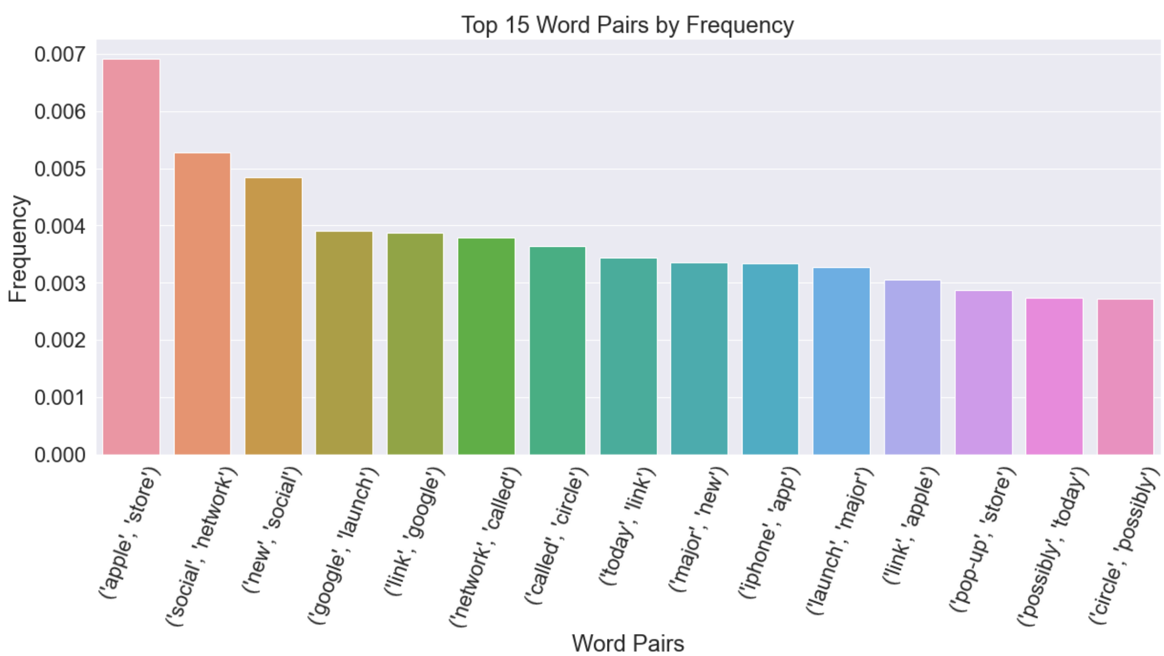
In [96]:
```
bigram_measures = nltk.collocations.BigramAssocMeasures()
tweet_finder = nltk.BigramCollocationFinder.from_words(clean_lemmatized_toker
tweets_scored = tweet_finder.score_ngrams(bigram_measures.raw_freq)
```

In [97]:
```
bigram1 = pd.DataFrame(tweets_scored, columns=['Words','Freq'])
bigram1.head()
```

Out[97]:

	Words	Freq
0	(apple, store)	0.006920
1	(social, network)	0.005277
2	(new, social)	0.004837
3	(google, launch)	0.003912
4	(link, google)	0.003877

In [98]:
```
fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=bigram1.head(15)['Words'], y=bigram1.head(15)['Freq'],
                 palette=palette)
ax.set(xlabel="Word Pairs",ylabel="Frequency")
plt.ticklabel_format(style='plain',axis='y')
plt.xticks(rotation=70)
plt.title('Top 15 Word Pairs by Frequency')
plt.show()
```



1.6 Keras NN Binary Classification

In [99]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import models, layers, optimizers
```

In [100]:
```
model = 0
```

1.6.1 Tokenize Upsampled Tweets

In [101]:
```
tweets = df_upsampled['Tweet']
tokenizer = Tokenizer(num_words=10000)
tokenizer.fit_on_texts(tweets)
sequences = tokenizer.texts_to_sequences(tweets)
print('sequences type: ' , type(sequences))
```

```
sequences type:  <class 'list'>
```

In [102]:
```
one_hot_results = tokenizer.texts_to_matrix(tweets, mode='binary')
print('one_hot_results type:', type(one_hot_results))
one_hot_results = np.asarray(one_hot_results)
```

```
one_hot_results type: <class 'numpy.ndarray'>
```

In [103]:
```
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

```
Found 4816 unique tokens.
```

In [104]:
```
print('Dimensions of our coded results:', np.shape(one_hot_results))
```

```
Dimensions of our coded results: (3500, 10000)
```

In [105]:
```
y = df_upsampled['Positive_Bin']
```

In [106]:
```
y = np.asarray(y)
```

In [107]:
```
print(y.shape)
print(one_hot_results.shape)
```

```
(3500,)
(3500, 10000)
```

In [108]:
```
print(len(y))
```

```
3500
```

In [109]:
```
import random
```

In [110]:
```
random.seed(42)
test_index = list(random.sample(range(1,3200),2000))

test = np.asarray(one_hot_results[test_index])
train = np.delete(one_hot_results, test_index, 0)

label_test = y[test_index]
label_train = np.delete(y, test_index, 0)

print('Test label shape:', np.shape(label_test))
print('Train label shape:', np.shape(label_train))
print('Test shape:', np.shape(test))
print('Train shape:', np.shape(train))
```

```
Test label shape: (2000,)
Train label shape: (1500,)
Test shape: (2000, 10000)
Train shape: (1500, 10000)
```

In [111]:
```
tokenizer.word_counts
```

Out[111]:
```
OrderedDict([('at', 1127),
             ('sxsw', 3630),
             ('tapworthy', 44),
             ('ipad', 1213),
             ('design', 89),
             ('headaches', 41),
             ('avoiding', 3),
             ('the', 1847),
             ('pitfalls', 3),
             ('of', 753),
             ('new', 357),
             ('challenges', 3),
             ('rt', 1000),
             ('mention', 2312),
             ('part', 12),
             ('journalsim', 5),
             ('is', 883),
             ('support', 15),
             ('democracy', 5),
```

In [112]:
```
print(type(X),X.shape)
print(type(y),y.shape)
```

```
<class 'pandas.core.series.Series'> (3500,)
<class 'numpy.ndarray'> (3500,)
```

1.6.2 Build Neural Network Model with Sigmoid Activation

In [113]:
```
# Initialize a sequential model
model = []
model = models.Sequential()
# Two layers with relu activation
model.add(layers.Dense(32, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['acc'])
```

In [114]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 32)                320032    
_________________________________________________________________
dense_1 (Dense)              (None, 16)                528       
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17        
=================================================================
Total params: 320,577
Trainable params: 320,577
Non-trainable params: 0
_________________________________________________________________
```

In [115]:
```
train.shape
```

Out[115]:
```
(1500, 10000)
```

In [116]:
```
label_train.shape
```

Out[116]:
```
(1500,)
```

1.6.3 Run Model

In [117]:
```
history = model.fit(train, label_train, batch_size=32, epochs=20, verbose=2,
                    validation_split=.2)
```

```
Epoch 1/20
38/38 - 3s - loss: 0.6429 - acc: 0.6700 - val_loss: 0.4062 - val_acc: 1.000
0
Epoch 2/20
38/38 - 0s - loss: 0.5088 - acc: 0.7217 - val_loss: 0.2940 - val_acc: 0.993
3
Epoch 3/20
38/38 - 0s - loss: 0.3224 - acc: 0.9067 - val_loss: 0.2452 - val_acc: 0.926
7
Epoch 4/20
38/38 - 0s - loss: 0.1455 - acc: 0.9650 - val_loss: 0.1764 - val_acc: 0.930
0
Epoch 5/20
38/38 - 0s - loss: 0.0613 - acc: 0.9933 - val_loss: 0.1494 - val_acc: 0.933
3
Epoch 6/20
38/38 - 0s - loss: 0.0294 - acc: 0.9983 - val_loss: 0.1710 - val_acc: 0.916
7
Epoch 7/20
38/38 - 0s - loss: 0.0161 - acc: 1.0000 - val_loss: 0.1996 - val_acc: 0.906
7
Epoch 8/20
38/38 - 0s - loss: 0.0100 - acc: 1.0000 - val_loss: 0.1820 - val_acc: 0.910
0
Epoch 9/20
38/38 - 0s - loss: 0.0067 - acc: 1.0000 - val_loss: 0.1726 - val_acc: 0.913
3
Epoch 10/20
38/38 - 0s - loss: 0.0049 - acc: 1.0000 - val_loss: 0.1868 - val_acc: 0.913
3
Epoch 11/20
38/38 - 0s - loss: 0.0037 - acc: 1.0000 - val_loss: 0.1816 - val_acc: 0.913
3
Epoch 12/20
38/38 - 0s - loss: 0.0029 - acc: 1.0000 - val_loss: 0.1857 - val_acc: 0.913
3
Epoch 13/20
38/38 - 0s - loss: 0.0023 - acc: 1.0000 - val_loss: 0.1990 - val_acc: 0.913
3
Epoch 14/20
38/38 - 0s - loss: 0.0019 - acc: 1.0000 - val_loss: 0.1935 - val_acc: 0.913
3
Epoch 15/20
38/38 - 0s - loss: 0.0016 - acc: 1.0000 - val_loss: 0.2002 - val_acc: 0.913
3
Epoch 16/20
38/38 - 0s - loss: 0.0014 - acc: 1.0000 - val_loss: 0.2055 - val_acc: 0.910
0
Epoch 17/20
38/38 - 0s - loss: 0.0012 - acc: 1.0000 - val_loss: 0.2031 - val_acc: 0.913
3
Epoch 18/20
38/38 - 0s - loss: 0.0010 - acc: 1.0000 - val_loss: 0.2042 - val_acc: 0.910
0
```

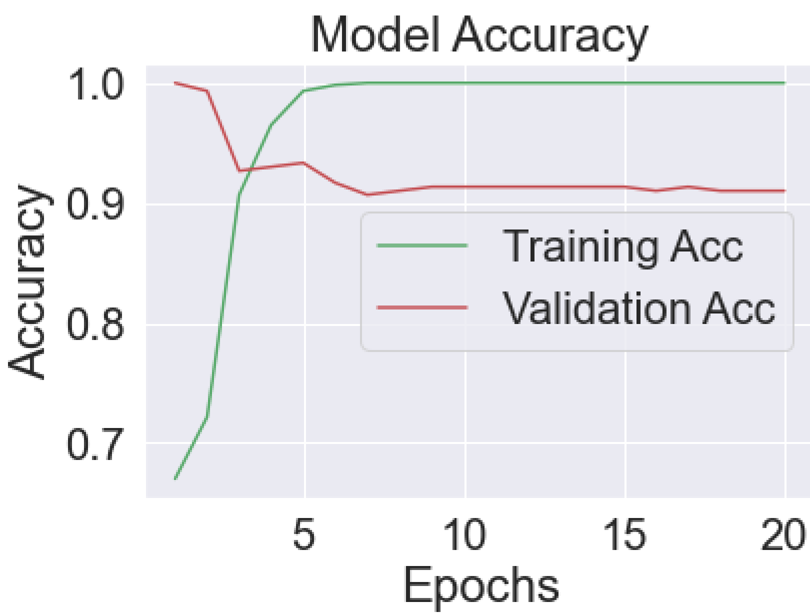
```
Epoch 19/20
38/38 - 0s - loss: 8.8706e-04 - acc: 1.0000 - val_loss: 0.2066 - val_acc:
0.9100
Epoch 20/20
38/38 - 0s - loss: 7.8655e-04 - acc: 1.0000 - val_loss: 0.2104 - val_acc:
0.9100
```

1.6.4 Training and Validation Graphs

In [118]:

```python
history_dict = history.history
loss_values = history_dict['loss']
loss_valid = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'g', label='Training Loss')
plt.plot(epochs, loss_valid, 'r', label='Validation Loss')
plt.title('Training Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

In [119]:
```
# Plot the training accuracy vs the number of epochs

acc_values = history_dict['acc']
acc_valid = history_dict['val_acc']

plt.figure()

plt.plot(epochs, acc_values, 'g', label='Training Acc')
plt.plot(epochs, acc_valid, 'r', label='Validation Acc')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc='right')
plt.show()
```



1.7 NLP using Word2Vec

In [120]:
```
from nltk import word_tokenize
```

1.7.1 Tokenize Tweets

In [121]:
```
data = df_upsampled['Tweet'].map(word_tokenize)
```

In [122]:
```
data[:10]
```

Out[122]:
```
1749    [At, #, sxsw, #, tapworthy, iPad, Design, Head...
6436    [RT, @, mention, Part, of, Journalsim, is, the...
3838    [Fuck, the, iphone, !, RT, @, mention, New, #,...
1770    [#, SXSW, 2011, :, Novelty, of, iPad, news, ap...
1062    [New, #, SXSW, rule, :, no, more, ooing, and, ...
324     [Overheard, at, #, sxsw, interactive, :, &, qu...
1944    [#, virtualwallet, #, sxsw, no, NFC, in, #, ip...
7201    [#, SXSW, a, tougher, crowd, than, Colin, Quin...
3159    [Why, is, wifi, working, on, my, laptop, but, ...
4631    [Is, starting, to, think, my, #, blackberry, i...
Name: Tweet, dtype: object
```

1.7.2 Create Word2Vec Model

In [123]:
```
model_W2V = Word2Vec(data, size =100, window=5, min_count=1, workers=4)
```

```
2020-12-26 12:20:25,468 : INFO : collecting all words and their counts
2020-12-26 12:20:25,469 : INFO : PROGRESS: at sentence #0, processed 0 wo
rds, keeping 0 word types
2020-12-26 12:20:25,497 : INFO : collected 5920 word types from a corpus
of 86715 raw words and 3500 sentences
2020-12-26 12:20:25,499 : INFO : Loading a fresh vocabulary
2020-12-26 12:20:25,521 : INFO : effective_min_count=1 retains 5920 uniqu
e words (100% of original 5920, drops 0)
2020-12-26 12:20:25,522 : INFO : effective_min_count=1 leaves 86715 word
corpus (100% of original 86715, drops 0)
2020-12-26 12:20:25,546 : INFO : deleting the raw counts dictionary of 59
20 items
2020-12-26 12:20:25,547 : INFO : sample=0.001 downsamples 52 most-common
words
2020-12-26 12:20:25,548 : INFO : downsampling leaves estimated 56808 word
corpus (65.5% of prior 86715)
2020-12-26 12:20:25,570 : INFO : estimated required memory for 5920 words
and 100 dimensions: 7696000 bytes
2020-12-26 12:20:25,571 : INFO : resetting layer weights
```

In [124]:
```
model_W2V.train(data,total_examples=model_W2V.corpus_count, epochs=10)
```

```
2020-12-26 12:20:27,403 : WARNING : Effective 'alpha' higher than previou
s training cycles
2020-12-26 12:20:27,404 : INFO : training model with 4 workers on 5920 vo
cabulary and 100 features, using sg=0 hs=0 sample=0.001 negative=5 window
=5
2020-12-26 12:20:27,472 : INFO : worker thread finished; awaiting finish
of 3 more threads
2020-12-26 12:20:27,480 : INFO : worker thread finished; awaiting finish
of 2 more threads
2020-12-26 12:20:27,484 : INFO : worker thread finished; awaiting finish
of 1 more threads
2020-12-26 12:20:27,487 : INFO : worker thread finished; awaiting finish
of 0 more threads
2020-12-26 12:20:27,488 : INFO : EPOCH - 1 : training on 86715 raw words
(56685 effective words) took 0.1s, 777797 effective words/s
2020-12-26 12:20:27,555 : INFO : worker thread finished; awaiting finish
of 3 more threads
2020-12-26 12:20:27,560 : INFO : worker thread finished; awaiting finish
of 2 more threads
```

In [125]:
```
wv = model_W2V.wv
```

In [126]:
```
wv.most_similar(positive='phone')
```

```
2020-12-26 12:20:28,238 : INFO : precomputing L2-norms of word weight vecto
rs
```

Out[126]:
```
[('moment-it', 0.9629813432693481),
 ('website', 0.9612559676170349),
 ('cases', 0.9578118324279785),
 ('since', 0.9564352035522461),
 ('j.mp/i41H53', 0.9556514024734497),
 ('dawdled', 0.9553717374801636),
 ('curse', 0.9548037648200989),
 ('words', 0.952925980091095),
 ('correcting', 0.9528586268424988),
 ('makes', 0.9525998830795288)]
```

In [127]:
```
wv['help']
```

Out[127]:
```
array([-0.03950676, -0.00265055, -0.2662849 , -0.4357826 ,  0.14867908,
        0.05602373, -0.13270333, -0.08401874, -0.05973308,  0.24611497,
        0.07503323,  0.15826945, -0.10546511, -0.25136733, -0.06458717,
        0.01225393, -0.01951106,  0.04807621, -0.13063331,  0.22152302,
       -0.4299225 , -0.20816697, -0.00807228, -0.17951213,  0.10377222,
        0.18258122, -0.08413679,  0.02349433, -0.04994519, -0.20971392,
       -0.00896185,  0.04977934, -0.22114964, -0.14219803,  0.18960081,
        0.00212619, -0.03621757,  0.14227545,  0.13329546, -0.22409889,
       -0.21578036, -0.00175644,  0.0667988 , -0.2752793 ,  0.14503188,
        0.14732912,  0.1975135 ,  0.4909825 , -0.04347203, -0.276607  ,
        0.20231214,  0.11232787,  0.13879468, -0.17216432, -0.06340772,
        0.17423737,  0.02715456, -0.00781501, -0.09893012,  0.10824313,
       -0.07071834, -0.10942367,  0.60323966,  0.11959276, -0.2515164 ,
       -0.04221073,  0.40175006, -0.21577579, -0.0278269 , -0.06996075,
        0.00589464, -0.25817883,  0.28745607,  0.04088598,  0.04244207,
        0.2736217 , -0.0707499 ,  0.02043922, -0.10660829,  0.17418425,
        0.0966424 ,  0.0406205 , -0.03688242,  0.08909915, -0.09917287,
        0.252028  ,  0.02200035, -0.1699533 ,  0.03755933, -0.2036003 ,
       -0.12875772,  0.19058114,  0.01087331,  0.01527689, -0.22210686,
        0.20285198, -0.00462554,  0.13788652, -0.32885122, -0.17243174],
      dtype=float32)
```

In [128]:
```
wv.vectors
```

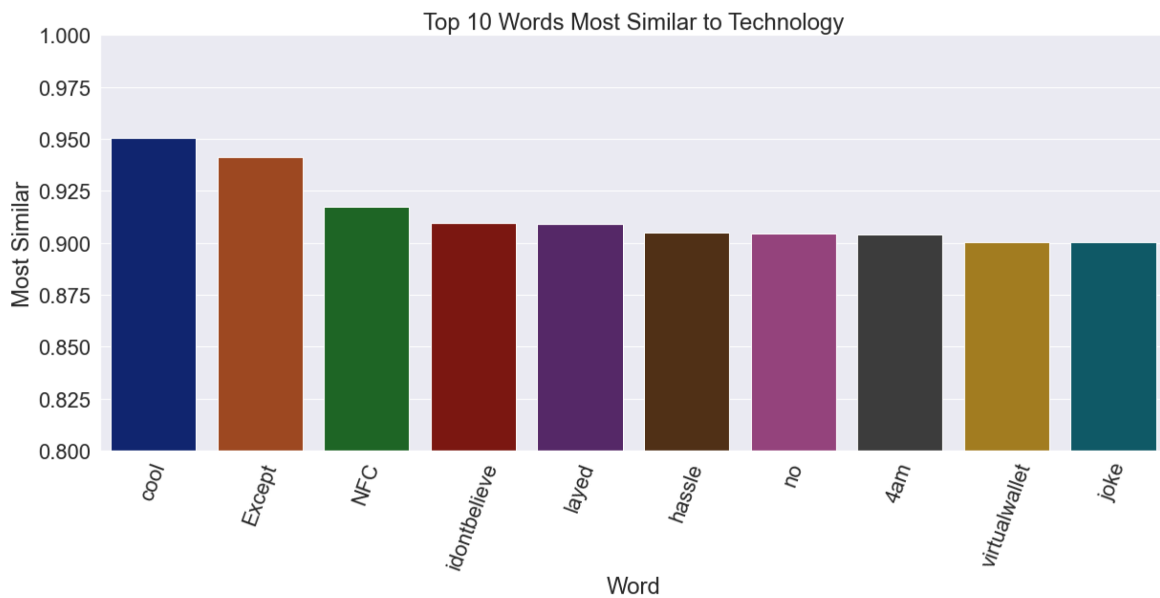
Out[128]:
```
array([[-0.14147308,  0.08278475, -0.86508656, ..., -0.24474262,
        -0.26447383,  0.06571927],
       [-0.0800463 , -0.19442783, -0.6667908 , ...,  0.1818839 ,
        -0.7173448 , -0.17950253],
       [-0.06608651, -0.33522293, -0.6358343 , ..., -0.47667727,
         1.0509689 ,  0.6043021 ],
       ...,
       [-0.01232286,  0.00259262, -0.04775942, ...,  0.01736588,
        -0.04775475, -0.00382941],
       [-0.01004897, -0.01281031,  0.00969365, ..., -0.01574854,
        -0.00828687,  0.00536801],
       [-0.01449329, -0.00127284, -0.01279595, ...,  0.015271  ,
        -0.0317597 , -0.02130941]], dtype=float32)
```

In [129]:
```
df_tech = pd.DataFrame(wv.most_similar(positive=['technology']))
```

1.7.3 Most Similar Words

In [130]:

```
fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=df_tech.head(10)[0], y=df_tech.head(10)[1],
                 palette=palette)
ax.set(xlabel="Word",ylabel="Most Similar")
plt.ticklabel_format(style='plain',axis='y')
plt.ylim(.8,1)
plt.xticks(rotation=70)
plt.title('Top 10 Words Most Similar to Technology')
plt.show()
```



In [131]:

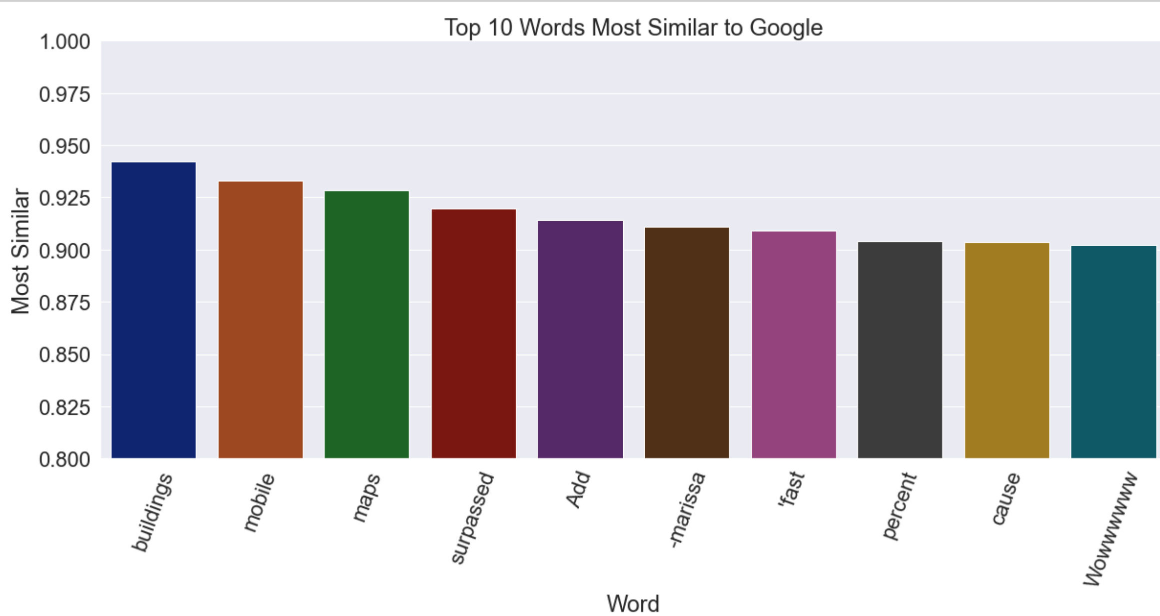
```
df_google = pd.DataFrame(wv.most_similar(positive=['google']))
df_google
```

Out[131]:

	0	1
0	buildings	0.942428
1	mobile	0.932969
2	maps	0.928591
3	surpassed	0.919620
4	Add	0.914240
5	-marissa	0.911062
6	'fast	0.909336
7	percent	0.904026
8	cause	0.903418
9	Wowwwwww	0.902320

In [132]:

```
fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=df_google.head(10)[0], y=df_google.head(10)[1],
                 palette=palette)
ax.set(xlabel="Word",ylabel="Most Similar")
plt.ticklabel_format(style='plain',axis='y')
plt.ylim(.8,1)
plt.xticks(rotation=70)
plt.title('Top 10 Words Most Similar to Google')
plt.show()
```

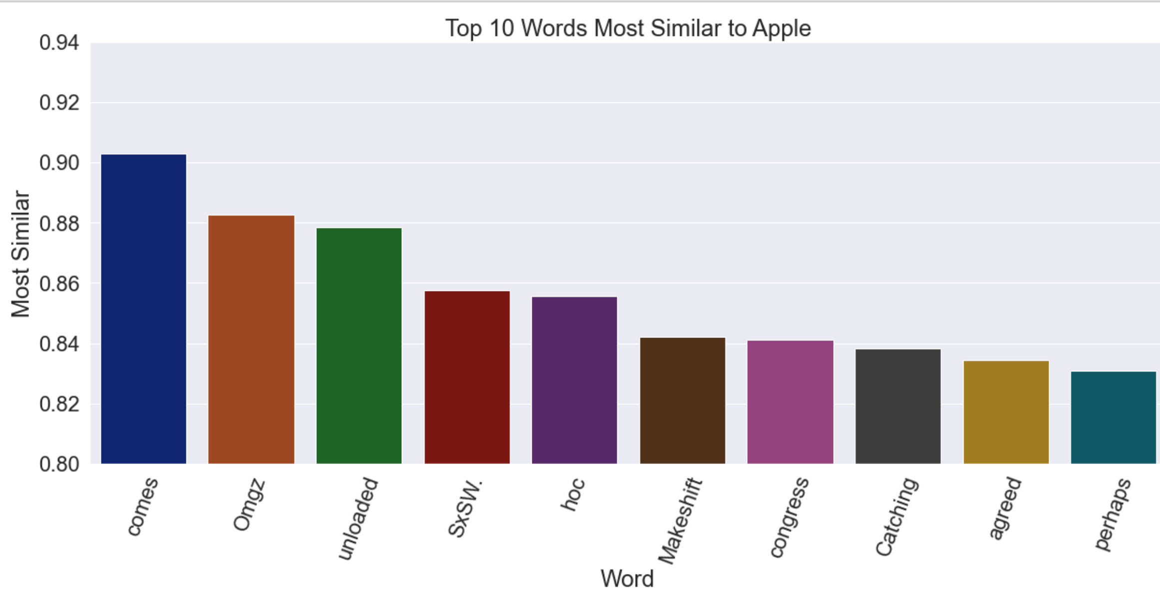


In [133]:
```
df_apple = pd.DataFrame(wv.most_similar(positive=['apple']))
df_apple
```

Out[133]:

	0	1
0	comes	0.902811
1	Omgz	0.882786
2	unloaded	0.878483
3	SxSW.	0.857545
4	hoc	0.855577
5	Makeshift	0.842251
6	congress	0.841276
7	Catching	0.838146
8	agreed	0.834300
9	perhaps	0.830786

In [134]:
```
fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=df_apple.head(10)[0], y=df_apple.head(10)[1], palette=pale
ax.set(xlabel="Word",ylabel="Most Similar")
plt.ticklabel_format(style='plain',axis='y')
plt.ylim(.8,.94)
plt.xticks(rotation=70)
plt.title('Top 10 Words Most Similar to Apple')
plt.show()
```



In [135]:
```
import nltk
nltk.download('vader_lexicon')

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import random
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from sklearn import preprocessing
from keras.preprocessing.text import Tokenizer
from keras import models
from keras import layers
from keras import optimizers
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\josep\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

1.8 Keras NN Multiple Classification

In [136]:
```
df = pd.read_csv('Tweet.csv')
df_up = pd.read_csv('Upsampled.csv')
```

In [137]:
```
df = df.drop(columns='Unnamed: 0')
```

In [138]:
```
df.head(5) # normal
```

Out[138]:

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

In [139]:
```
df_up = df_up.drop(columns='Unnamed: 0')
```

In [140]:
```
df_up.head(5) # upsampled for increased number of negative tweets
```

Out[140]:

	Tweet	Platform	Emotion	Positive_Bin
0	At #sxsw #tapworthy iPad Design Headaches - av...	iPad	Negative emotion	0
1	RT @mention Part of Journalsim is the support ...	NaN	Negative emotion	0
2	Fuck the iphone! RT @mention New #UberSocial f...	iPhone	Negative emotion	0
3	#SXSW 2011: Novelty of iPad news apps fades fa...	iPad	Negative emotion	0
4	New #SXSW rule: no more ooing and ahing over y...	iPad	Negative emotion	0

In [141]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3548 entries, 0 to 3547
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Tweet         3548 non-null   object
 1   Platform      3191 non-null   object
 2   Emotion       3548 non-null   object
 3   Positive_Bin  3548 non-null   int64 
dtypes: int64(1), object(3)
memory usage: 111.0+ KB
```

In [142]:
```
df_up.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3500 entries, 0 to 3499
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Tweet         3500 non-null   object
 1   Platform      3171 non-null   object
 2   Emotion       3500 non-null   object
 3   Positive_Bin  3500 non-null   int64 
dtypes: int64(1), object(3)
memory usage: 109.5+ KB
```

In [143]:
```
df_up['Positive_Bin'].value_counts()
```

Out[143]:
```
1    2500
0    1000
Name: Positive_Bin, dtype: int64
```

1.8.1 VADER Sentiment Analysis

In [144]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

In [145]:
```
sid = SentimentIntensityAnalyzer()
```

In [146]:
```
df_up['scores'] = df_up['Tweet'].apply(lambda review:sid.polarity_scores(revi
```

In [147]:
```
df_up['compound'] = df_up['scores'].apply(lambda d:d['compound'])
```

In [148]:
```
df_up['comp_score'] = df_up['compound'].apply(lambda score: 1
                                               if score >= 0 else 0)
```

In [149]:
```
df_up.head()
```

Out[149]:

	Tweet	Platform	Emotion	Positive_Bin	scores	compound	comp_score
0	At #sxsw #tapworthy iPad Design Headaches - av...	iPad	Negative emotion	0	{'neg': 0.153, 'neu': 0.764, 'pos': 0.083, 'co...	-0.2732	0
1	RT @mention Part of Journalsim is the support ...	NaN	Negative emotion	0	{'neg': 0.0, 'neu': 0.63, 'pos': 0.37, 'compou...	0.8796	1
2	Fuck the iphone! RT @mention New #UberSocial f...	iPhone	Negative emotion	0	{'neg': 0.166, 'neu': 0.834, 'pos': 0.0, 'comp...	-0.5848	0
3	#SXSW 2011: Novelty of iPad news apps fades fa...	iPad	Negative emotion	0	{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound...	0.0000	1
4	New #SXSW rule: no more ooing and ahing over y...	iPad	Negative emotion	0	{'neg': 0.083, 'neu': 0.83, 'pos': 0.087, 'com...	0.0258	1

In [150]:
```
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
```

In [151]:
```
acc_score = accuracy_score(df_up['Positive_Bin'],df_up['comp_score'])
```

In [152]:
```
print('Accuracy Score: ', "{:.3f}".format(acc_score*100),"%")
```

```
Accuracy Score:  75.371 %
```

In [153]:
```
print(classification_report(df_up['Positive_Bin'],df_up['comp_score']))
```

```
              precision    recall  f1-score   support

           0       0.61      0.39      0.47      1000
           1       0.79      0.90      0.84      2500

    accuracy                           0.75      3500
   macro avg       0.70      0.64      0.66      3500
weighted avg       0.74      0.75      0.73      3500
```

1.8.2 VADER Confusion Matrix

In [154]:
```
confusion_matrix(df_up['Positive_Bin'],df_up['comp_score'])
```

Out[154]:
```
array([[ 389,  611],
       [ 251, 2249]], dtype=int64)
```

VADER doesn't do a great job of correctly classifying tweet sentiment, with 611 false positive tweets that are actually negative

In [155]:
```
full_df = pd.read_csv('Full_DF')
```

In [156]:
```
full_df.head()
```

Out[156]:

	Unnamed: 0	Tweet	Platform	Emotion	Uncertain	Negative	No Emotion	Positive
0	0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative emotion	0	1	0	0
1	1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive emotion	0	0	0	1
2	2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive emotion	0	0	0	1
3	3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Negative emotion	0	1	0	0
4	4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Positive emotion	0	0	0	1

In [157]:
```
full_df = full_df.drop(columns='Unnamed: 0')
```

In [158]:
```
full_df.head(10)
full_df = full_df.dropna()
```

1.8.3 Tokenize Tweets

In [159]:
```
tweets = full_df['Tweet']
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(tweets)
sequences = tokenizer.texts_to_sequences(tweets)
print('sequences type: ' , type(sequences))
```

```
sequences type:  <class 'list'>
```

In [160]:
```
one_hot_results = tokenizer.texts_to_matrix(tweets, mode='binary')
print('one_hot_results type:', type(one_hot_results))
```

```
one_hot_results type: <class 'numpy.ndarray'>
```

In [161]:
```
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

```
Found 5963 unique tokens.
```

In [162]:
```
# Our coded data
print('Dimensions of our coded results:', np.shape(one_hot_results))
```

```
Dimensions of our coded results: (3291, 5000)
```

In [163]:
```
print(y.shape)
print(one_hot_results.shape)
```

```
(3500,)
(3291, 5000)
```

In [164]:

```
emotion = full_df['Emotion']

# Initialize
le = preprocessing.LabelEncoder()
le.fit(emotion)
print('Original class labels:')
print(list(le.classes_))
print('\n')
emotion_cat = le.transform(emotion)

# If you wish to retrieve the original descriptive labels post production
# list(le.inverse_transform([0, 1, 3, 3, 0, 6, 4]))

print('New product labels:')
print(emotion_cat)
print('\n')

# Each row will be all zeros except for the category for that observation
print('One hot labels; 4 binary columns, one for each of the categories.')
product_onehot = to_categorical(emotion_cat)
print(product_onehot)
print('\n')

print('One hot labels shape:')
print(np.shape(product_onehot))
```

```
Original class labels:
["I can't tell", 'Negative emotion', 'No emotion toward brand or product',
'Positive emotion']


New product labels:
[1 3 3 ... 1 3 3]


One hot labels; 4 binary columns, one for each of the categories.
[[0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 0. 1.]
 ...
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 0. 1.]]


One hot labels shape:
(3291, 4)
```

In [165]:
```
random.seed(42)
test_index = random.sample(range(1,3200), 1500)

test = one_hot_results[test_index]
train = np.delete(one_hot_results, test_index, 0)

label_test = product_onehot[test_index]
label_train = np.delete(product_onehot, test_index, 0)

print('Test label shape:', np.shape(label_test))
print('Train label shape:', np.shape(label_train))
print('Test shape:', np.shape(test))
print('Train shape:', np.shape(train))
```

```
Test label shape: (1500, 4)
Train label shape: (1791, 4)
Test shape: (1500, 5000)
Train shape: (1791, 5000)
```

1.8.4 Build Neural Network Model

In [166]:
```
from keras.layers import Input, Dense, LSTM, Embedding
from keras.layers import Dropout, Activation, Bidirectional, GlobalMaxPool1D
from keras.models import Sequential
```

In [167]:
```
# Initialize  and build a sequential model
model = models.Sequential()
# Two layers with relu activation
model.add(layers.Dense(50, activation='relu', input_shape=(5000,)))
model.add(layers.Dense(25, activation='relu'))
model.add(layers.Dense(4, activation='softmax'))
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['acc'])
```

1.8.5 Run Model

In [168]:

```
history = model.fit(train,
                    label_train,
                    epochs=20,
                    batch_size=32,
                    validation_split=.2)
```

```
Epoch 1/20
45/45 [==============================] - 1s 11ms/step - loss: 1.0344 - ac
c: 0.7092 - val_loss: 0.6501 - val_acc: 0.8162
Epoch 2/20
45/45 [==============================] - 0s 4ms/step - loss: 0.4771 - ac
c: 0.8258 - val_loss: 0.6118 - val_acc: 0.8134
Epoch 3/20
45/45 [==============================] - 0s 4ms/step - loss: 0.3744 - ac
c: 0.8513 - val_loss: 0.5857 - val_acc: 0.8329
Epoch 4/20
45/45 [==============================] - 0s 4ms/step - loss: 0.1880 - ac
c: 0.9503 - val_loss: 0.5813 - val_acc: 0.8301
Epoch 5/20
45/45 [==============================] - 0s 4ms/step - loss: 0.1328 - ac
c: 0.9564 - val_loss: 0.6202 - val_acc: 0.8245
Epoch 6/20
45/45 [==============================] - ETA: 0s - loss: 0.0701 - acc: 0.
980 - 0s 4ms/step - loss: 0.0700 - acc: 0.9801 - val_loss: 0.6446 - val_a
cc: 0.8217
Epoch 7/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0489 - ac
c: 0.9897 - val_loss: 0.7002 - val_acc: 0.8134
Epoch 8/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0389 - ac
c: 0.9907 - val_loss: 0.7646 - val_acc: 0.8357
Epoch 9/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0274 - ac
c: 0.9906 - val_loss: 0.7917 - val_acc: 0.8384
Epoch 10/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0165 - ac
c: 0.9969 - val_loss: 0.8299 - val_acc: 0.8329
Epoch 11/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0217 - ac
c: 0.9921 - val_loss: 0.8507 - val_acc: 0.8301
Epoch 12/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0181 - ac
c: 0.9962 - val_loss: 0.8942 - val_acc: 0.8384
Epoch 13/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0103 - ac
c: 0.9971 - val_loss: 0.9068 - val_acc: 0.8357
Epoch 14/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0070 - ac
c: 0.9989 - val_loss: 0.9070 - val_acc: 0.8106
Epoch 15/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0141 - ac
c: 0.9977 - val_loss: 0.9571 - val_acc: 0.8329
Epoch 16/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0070 - ac
c: 0.9986 - val_loss: 0.9649 - val_acc: 0.8357
Epoch 17/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0052 - ac
```

```
c: 0.9976 - val_loss: 0.9751 - val_acc: 0.8329
Epoch 18/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0071 - ac
c: 0.9961 - val_loss: 0.9873 - val_acc: 0.8357
Epoch 19/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0078 - ac
c: 0.9985 - val_loss: 1.0141 - val_acc: 0.8357
Epoch 20/20
45/45 [==============================] - 0s 4ms/step - loss: 0.0068 - ac
c: 0.9969 - val_loss: 1.0094 - val_acc: 0.8245
```

In [169]:

```
history_dict = history.history
```

In [170]:

```
history_dict.keys()
```

Out[170]:

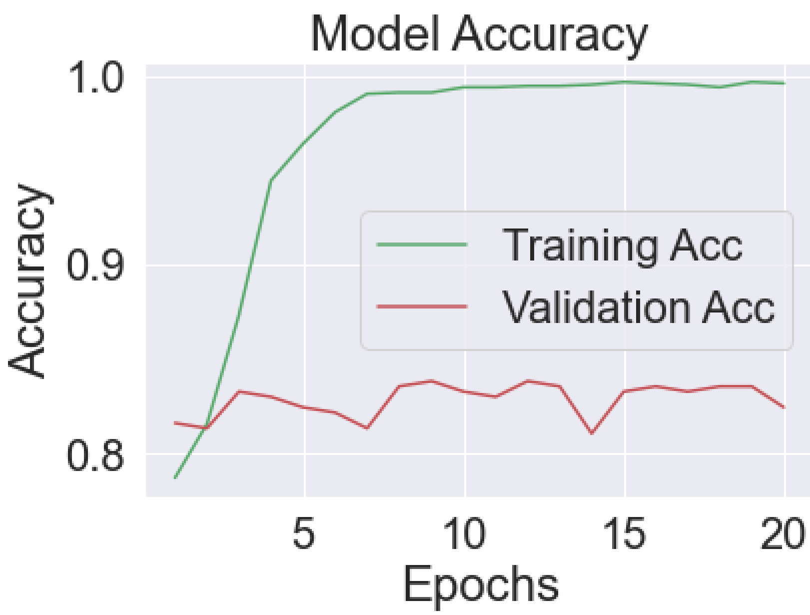
```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

1.8.6 Training and Validation Graphs

In [171]:

```
history_dict = history.history
loss_values = history_dict['loss']
loss_valid = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)

plt.plot(epochs, loss_values, 'g', label='Training Loss')
plt.plot(epochs, loss_valid, 'r', label='Validation Loss')
plt.title('Training Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

In [172]:
```
# Plot the training accuracy vs the number of epochs

acc_values = history_dict['acc']
acc_valid = history_dict['val_acc']

plt.figure()

plt.plot(epochs, acc_values, 'g', label='Training Acc')
plt.plot(epochs, acc_valid, 'r', label='Validation Acc')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc='right')
plt.show()
```



In [173]:
```
# Output (probability) predictions for the test set
y_hat_test = model.predict(test)
```

In [174]:
```
# Print the loss and accuracy for the training set
results_train = model.evaluate(train, label_train)
results_train
```

```
56/56 [==============================] - 0s 2ms/step - loss: 0.2070 - acc:
0.9631
```

Out[174]:
```
[0.20703548192977905, 0.9631490707397461]
```

In [175]:
```
results_test = model.evaluate(test, label_test)
results_test # model predicts on the test data with almost 84% accuracy.
```

```
47/47 [==============================] - 0s 1ms/step - loss: 0.8442 - acc:
0.8340
```

Out[175]:
```
[0.8441706299781799, 0.8339999914169312]
```

1.9 Question 1 and Recommendation

1.9.1 In tweets targeting either the iPhone or Android phones, which product is more often the subject of negatively charged emotions?

In [176]:
```
df_neg = pd.read_csv('Full_DF')
df_neg = df_neg.drop(columns='Unnamed: 0')
```

In [177]:
```
df_neg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9093 entries, 0 to 9092
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Tweet       9092 non-null   object
 1   Platform    3291 non-null   object
 2   Emotion     9093 non-null   object
 3   Uncertain   9093 non-null   int64 
 4   Negative    9093 non-null   int64 
 5   No Emotion  9093 non-null   int64 
 6   Positive    9093 non-null   int64 
dtypes: int64(4), object(3)
memory usage: 497.4+ KB
```

In [178]:
```
df_grouped = df_neg.groupby(by=df_neg['Platform']).sum()
```

In [179]:
```
df_grouped.index
```

Out[179]:
```
Index(['Android', 'Android App', 'Apple', 'Google',
       'Other Apple product or service', 'Other Google product or service',
       'iPad', 'iPad or iPhone App', 'iPhone'],
      dtype='object', name='Platform')
```

In [180]:
```
df_grouped
```

Out[180]:

Platform	Uncertain	Negative	No Emotion	Positive
Android	0	8	1	69
Android App	0	8	1	72
Apple	2	95	21	543
Google	1	68	15	346
Other Apple product or service	0	2	1	32
Other Google product or service	1	47	9	236
iPad	4	125	24	793
iPad or iPhone App	0	63	10	397
iPhone	1	103	9	184

In [181]:
```
# separate tweets
df_android = df_grouped.loc[df_grouped.index =='Android']
df_iphone = df_grouped.loc[df_grouped.index =='iPhone']
```

In [182]:
```
df_android
```

Out[182]:

Platform	Uncertain	Negative	No Emotion	Positive
Android	0	8	1	69

In [183]:
```
percent_negative_android_tweets = df_android['Negative']/sum(df_android['Posi
```

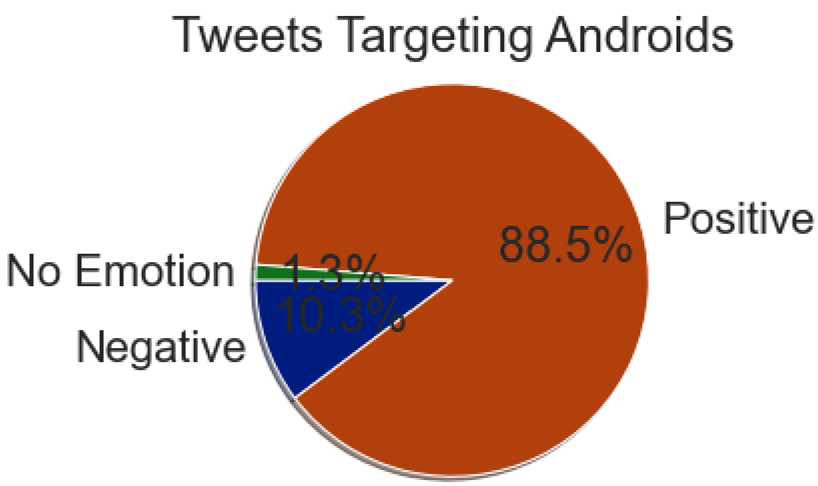
In [184]:
```
print("Percentage of tweets targeting Android phones that are negative: {:.3f
```

```
Percentage of tweets targeting Android phones that are negative: 10.390 %
```

In [185]:
```
labels1 = 'Negative', 'Positive', 'No Emotion'
sizes1 = [8, 69, 1]
```

1.9.2 Negative Tweets

In [186]:
```
fig1, ax1 = plt.subplots()
ax1.pie(sizes1, labels=labels1, autopct='%1.1f%%',
        shadow=True, startangle=180)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.title("Tweets Targeting Androids")

plt.show()
```



In [187]:
```
df_iphone
```

Out[187]:

	Uncertain	Negative	No Emotion	Positive
Platform				
iPhone	1	103	9	184

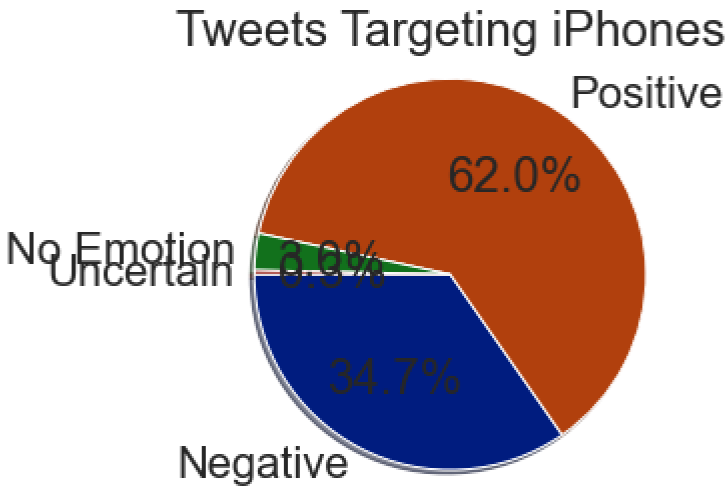
In [188]:
```
percent_negative_iphone_tweets = df_iphone['Negative']/sum(df_iphone['Negativ
```

In [189]:
```
print("Percentage of tweets targeting iPhones that are negative: {:.3f}".form
```

```
Percentage of tweets targeting iPhones that are negative: 35.889 %
```

In [190]:
```
sizes2 = [103, 184, 9, 1]
labels2 = 'Negative', 'Positive', 'No Emotion', 'Uncertain'
```

In [191]:
```
fig1, ax1 = plt.subplots()
ax1.pie(sizes2, labels=labels2, autopct='%1.1f%%',
        shadow=True, startangle=180)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.title("Tweets Targeting iPhones")

plt.show()
```



1.9.3 Recommendation

In creating your phone, more users may want the option to have a more customizable user interface which the Android provides. We will need to look into more detail about what negative words users are including in their negative tweets that target iPhones to specifically determine users' complaints.

1.10 Question 2 and Recommendation

1.10.1 What words are most common in negative tweets about iPhones and Android phones?

1.10.2 Negative Android Sentiment

In [192]:
```
# collect all negative tweets for each product
df_neg_android = df_neg.loc[df_neg['Platform'] =='Android']
df_neg_iphone = df_neg.loc[df_neg['Platform'] =='iPhone']
```

In [193]:
```
df_neg_android = df_neg_android.loc[df_neg_android['Negative'] == 1]
```

In [194]:
```
df_neg_android # tweets about Android that are negative - create bag of words
```

Out[194]:

	Tweet	Platform	Emotion	Uncertain	Negative	No Emotion	Positive
350	they took away the lego pit but replaced it wi...	Android	Negative emotion	0	1	0	0
1940	Why does all the #Android meetups here in #Aus...	Android	Negative emotion	0	1	0	0
1999	@mention Android needs a way to group apps lik...	Android	Negative emotion	0	1	0	0
3389	Lunch with @mention at #CNNGrill. View from th...	Android	Negative emotion	0	1	0	0
4865	Excited to meet the @mention at #sxsw so I can...	Android	Negative emotion	0	1	0	0
8053	Spending some time this morning resetting my a...	Android	Negative emotion	0	1	0	0
8258	Is it just me or has the @mention client for A...	Android	Negative emotion	0	1	0	0
8801	Auntie's voxpop of popular #sxsw apps is wort...	Android	Negative emotion	0	1	0	0

In [195]:
```
corpus_android = list(df_neg_android['Tweet'])
```

In [196]:
```
corpus_android[:10] # entirety of negative android tweets
```

Out[196]:
```
['they took away the lego pit but replaced it with a recharging station ;)
#sxsw and i might check prices for an iphone - crap samsung android',
 "Why does all the #Android meetups here in #Austin are when I'm at work. W
ell at least there is the PS meetup #sxsw",
 '@mention Android needs a way to group apps like you can now do with iPad/
iPod. #SXSW #hhrs',
 'Lunch with @mention at #CNNGrill. View from the HTML5 dev trenches: Andro
id is painful, iOS is sleek (for what @mention is doing) #sxsw',
 'Excited to meet the @mention at #sxsw so I can show them my Sprint Galaxy
S still running Android 2.1.   #fail',
 'Spending some time this morning resetting my android phone.  First day of
#sxsw was too much for it.',
 'Is it just me or has the @mention client for Android gotten really buggy
lately? #SXSW to blame?',
 "Auntie's voxpop of  popular #sxsw apps is worth a watch: {link} Not many
Android phones on view."]
```

In [197]:
```
# tokenize
android_tokens = word_tokenize(','.join(str(v) for v in corpus_android))

# remove stopwords
stopped_android_tokens = [word.lower() for word in android_tokens if word.low
                          not in stopword_list]
```

In [198]:
```
freq = FreqDist(stopped_android_tokens)
```

In [199]:
```
freq.most_common(25)
```

Out[199]:
```
[('android', 8),
 ('apps', 2),
 ('view', 2),
 ('took', 1),
 ('away', 1),
 ('lego', 1),
 ('pit', 1),
 ('replaced', 1),
 ('recharging', 1),
 ('station', 1),
 ('check', 1),
 ('prices', 1),
 ('iphone', 1),
 ('crap', 1),
 ('samsung', 1),
 ('meetups', 1),
 ('austin', 1),
 ('work', 1),
 ('ps', 1),
 ('meetup', 1),
 ('needs', 1),
 ('way', 1),
 ('group', 1),
 ('like', 1),
 ('ipad/ipod', 1)]
```

1.10.3 Negative iPhone Sentiment

In [200]:
```
df_neg_iphone = df_neg_iphone.loc[df_neg_iphone['Negative'] == 1]
```

In [201]:
```
df_neg_iphone # tweets about iphone that are negative - create bag of words i
```

Out[201]:

	Tweet	Platform	Emotion	Uncertain	Negative	No Emotion	Positive
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative emotion	0	1	0	0
17	I just noticed DST is coming this weekend. How...	iPhone	Negative emotion	0	1	0	0
92	What !?!? @mention #SXSW does not provide iPh...	iPhone	Negative emotion	0	1	0	0
233	If iPhone alarms botch the timechange, how man...	iPhone	Negative emotion	0	1	0	0
236	I meant I also wish I at #SXSW #dyac stupid i...	iPhone	Negative emotion	0	1	0	0
...	...	...	...	...	...	...	...

In [202]:
```
corpus_iphone = list(df_neg_iphone['Tweet'])
```

In [203]:
```
corpus_iphone[:15]
```

Out[203]:
```
['.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it w
as dead!  I need to upgrade. Plugin stations at #SXSW.',
 'I just noticed DST is coming this weekend. How many iPhone users will be
an hour late at SXSW come Sunday morning? #SXSW #iPhone',
 "What !?!? @mention  #SXSW does not provide iPhone chargers?!?  I've chang
ed my mind about going next year!",
 "If iPhone alarms botch the timechange, how many #SXSW'ers freak? Late to
flights, missed panels, behind on bloody marys...",
 'I meant I also wish I  at #SXSW #dyac stupid iPhone!',
 'Overheard at #sxsw interactive: &quot;Arg! I hate the iphone! I want my b
lackberry back&quot; #shocked',
 "overheard at MDW (and I'll second it) &quot;halfway through my iPhone bat
tery already and I haven't even boarded the plane to #sxsw&quot; #amateurho
ur",
 "My iPhone battery can't keep up with my tweets! Thanks Apple.  #SXSW  #pr
ecommerce",
 'IPhone is dead. Find me on the secret batphone #sxsw.',
 'Austin is getting full, and #SXSW is underway.  I can tell because my iPh
one is an intermittent brick. #crowded',
 '.@mention I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it wa
s dead!  I need to upgrade. Plugin stations at #SXSW.',
 'my iPhone is overheating. why are there so many british sounding people i
n texas? #SXSW',
 'My iPhone is wilting under the stress of being at #sxsw.',
 'iPhone, I know this #SXSW week will be tough on your already-dwindling ba
ttery, but so help me Jeebus if you keep correcting my curse words.',
 "God, it's like being at #sxsw - have iMac, MacBook, iPhone and BlackBerry
all staring at me. Enough! Time to read a book - remember those?"]
```

In [204]:
```
# tokenize
iphone_tokens = word_tokenize(','.join(str(v) for v in corpus_iphone))

# remove stopwords
stopped_iphone_tokens = [word.lower() for word in iphone_tokens if word.lower
                          not in stopword_list]
```

In [205]:
```
freq = FreqDist(stopped_iphone_tokens)
```

In [206]:
```
freq.most_common(25)
```

Out[206]:
```
[('iphone', 104),
 ('quot', 22),
 ('battery', 15),
 ('amp', 10),
 ('blackberry', 8),
 ('link', 8),
 ('austin', 7),
 ('app', 7),
 ('users', 6),
 ('going', 6),
 ('time', 6),
 ('sxsw.', 5),
 ('like', 5),
 ('u', 5),
 ('good', 5),
 ('3g', 4),
 ('hour', 4),
 ('apple', 4),
 ('people', 4),
 ('know', 4),
 ('ipad', 4),
 ('t-mobile', 4),
 ('shit', 4),
 ('long', 4),
 ('technology', 4)]
```

1.10.4 Recommendation

The Android operating system was claimed to be buggy in addition to someone saying Android is painful and not sleek like Apple's iOS. Generally, users had less negative things to say as a percentage of total comments.

The iPhone was said to have failing battery or a battery charge that does not last long enough when the phone is in operation. Additionally, lack of signal became a problem in crowded areas but this is not typically a phone design issue but instead is an infrastructure problem.

Build a sleek phone with a simple to use Graphical User Interface. Have plenty of battery to power the phone for longer periods. Users would enjoy a feature like a backup battery, or a sleekly designed case that provides a full second charge without adding much volume.

1.11 Question 3 and Recommendation

1.11.1 What are some of the positive features commented about for both iPhones and Android phones?

In [207]:
```
df_pos = pd.read_csv('Full_DF')
```

In [208]:
```
df_pos_android = df_pos.loc[df_pos['Platform'] =='Android']
df_pos_iphone = df_pos.loc[df_pos['Platform'] =='iPhone']
```

In [209]:
```
df_pos_android = df_pos_android.loc[df_pos_android['Positive']==1]
df_pos_iphone = df_pos_iphone.loc[df_pos_iphone['Positive']==1]
```

1.11.2 Positive Android Sentiment

In [210]:
```
corpus_android = list(df_pos_android['Tweet'])
```

In [211]:
```
corpus_android[:20]
```

Out[211]:
```
['#SXSW is just starting, #CTIA is around the corner and #googleio is only
a hop skip and a jump from there, good time to be an #android fan',
 'Excited to meet the @samsungmobileus at #sxsw so I can show them my Sprin
t Galaxy S still running Android 2.1.   #fail',
 'This is a #WINNING picture #android #google #sxsw {link}',
 "I knew if I plied @mention with beer and stogies last night I'd weasel my
way into the Team Android party tonight. #success #SXSW.",
 'Alert the media.  I just saw the one and only Android tablet at #sxsw.  L
ike finding a needle in a haystack!  I also saw a Cr-48.',
 'Farooqui: Now about mobile. iOS, with Android catching up fast and will g
row more once they allow in-app purchasing. #gamesfortv #sxsw',
 'I need to play this game on my #android - #SXSW {link}',
 'Talked to some great developers at the Android meetup. Looking forward to
working with them. #sxsw #android #androidsxsw',
 "There are thousands of iPad 2's floating around Austin at #sxsw and I hav
e not seen even one single Android tablet.  Not even one. Zero.",
 'Woot! RT @mention First Android @mention disc {link} ... Market version c
oming soon! #SXSW',
 'Heard at #sxsw #Android is now the leading market share of smart phones i
n US. #getjarsxsw',
 'Quadroid = Qualcomm + Android just called the platform of the next decade
vs Wintel #sxsw #cloud',
 '{link} via @mention pretty neat database I must say.  does it work on my
#android we shall see. #sxsw #party #free',
 "@mention Android just got a big call out at #sxsw in they #gamelayer open
ing keynote. I knew you'd appreciate.",
 'Android party #sxsw (@mention Lustre Pearl Bar w/ 36 others) {link}',
 '@mention at Team Android party. @mention @mention just walked in. DL Appo
licious app &amp; enter to win free Nexus S! #androidsxsw #sxsw',
 'Piece of awesomeness: Arduino + android = Flaming skulls  {link} @mention
@mention #sxsw #smartthings',
 '@mention Congratulations on winning the Android award! :) #sxsw',
 '@mention crew ripped up Android party - thanks for having us Droid! {lin
k} #sxsw',
 'Great UI demo of @mention on @mention {link} #xoom #sxsw #android #tech #
tablet']
```

In [212]:
```
# tokenize
android_tokens = word_tokenize(','.join(str(v) for v in corpus_android))

# remove stopwords
stopped_android_tokens = [word.lower() for word in android_tokens if word.low
                          not in stopword_list]
```

In [213]:
```
freq = FreqDist(stopped_android_tokens)
```

In [214]:
```
freq.most_common(25)
```

Out[214]:
```
[('android', 71),
 ('link', 26),
 ('party', 12),
 ('team', 11),
 ('free', 9),
 ('lustre', 6),
 ('pearl', 6),
 ('amp', 6),
 ('new', 6),
 ('phone', 6),
 ('dev', 5),
 ('tablet', 4),
 ('need', 4),
 ('great', 4),
 ('meetup', 4),
 ('androidsxsw', 4),
 ('market', 4),
 ('win', 4),
 ('love', 4),
 ('details', 4),
 ('starting', 3),
 ('good', 3),
 ('fan', 3),
 ('excited', 3),
 ('beer', 3)]
```

1.11.3 Positive iPhone Sentiment

In [215]:
```
corpus_iphone = list(df_pos_iphone['Tweet'])
```

In [216]:
```
corpus_iphone[:20]
```

Out[216]:
```
["I love my @mention iPhone case from #Sxsw but I can't get my phone out of
it #fail",
 'Yai!!! RT @mention New #UberSocial for #iPhone now in the App Store inclu
des UberGuide to #SXSW sponsored by (cont) {link}',
 'Take that #SXSW ! RT @mention Major South Korean director gets $130,000 t
o make a movie entirely with his iPhone. {link}',
 'Behind on 100s of emails? Give them all 1 line iPhone composed replies. #
SXSW #protip',
 'Picked up a Mophie battery case 4 my iPhone in prep for #SXSW. Not luggin
g around a laptop &amp; only using my phone was a huge win last year.',
 "Do I need any more for #sxsw! ipad, iphone, laptop, dictaphone, vid.camer
a.... Wow! Love to  meet the REAL 'cerebellum' charged people:)",
 'My iPhone battery at 100%. #winning at #SXSW',
 'BEST SWAG EVER. Thanks @mention My charging iPhone thanks you, too. #SXSW
{link}',
 'Love that I have a MacBook, iPad, and iPhone with me at #sxsw this year.
One runs out of juice, and I can jump to the next.',
 'Holy cow! I just got hooked by Paolo and Alex with a backup charger for m
y iPhone! facebook.com/powermat #powermatteam #sxsw #thanks',
 'Holy cow! I just got hooked by Paolo and Alex with a backup charger for m
y iPhone! facebook.com/powermat #powermattteam #sxsw #thanks',
 "@mention  I'm beyond frustrated w/ @mention after this Samsung Moment run
around &amp; am leaving for ATT &amp; iPhone so I can enjoy #sxsw.",
 "Tim Soo's invisible instruments are jaw dropping. iPhone+Wii controller.
{link} #lovemusicapi #sxsw",
 'I fear no iphone + #att 3gs slowpoke network during #sxsw &amp; #sxswmusi
c.',
 'Check out iPhone Developer Meet Up at SXSW.\n{link} #SXSW',
 "&quot;the iPhone is a transient device used in short bursts; the iPad is
an 'after 8pm, on the couch' device.&quot; @mention #sxsw",
 '@mention  iPhone. Clearly. Positively. Happily. #SXSW',
 'Flipboard is developing an iPhone version, not Android, says @mention #sx
sw',
 "So {link} is part of my presentation at #SXSW so good thing it's crashing
now instead of then. Works best on iPhone/Android",
 'Loving my Morphie JuicePack today for a recharge of iPhone. So worth it.
#sxsw']
```

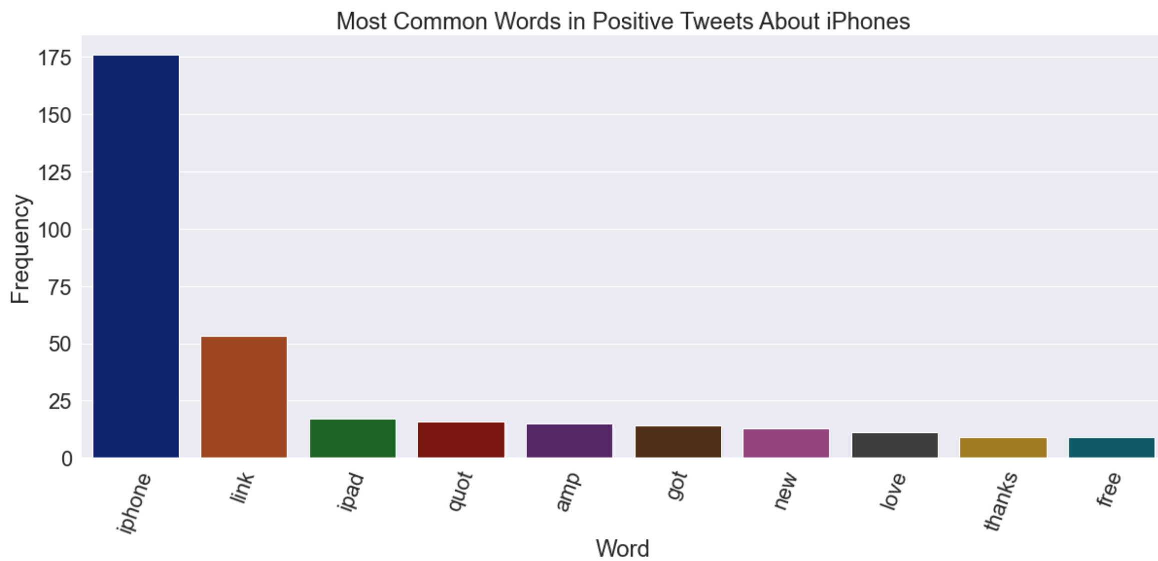
In [217]:
```
# tokenize
iphone_tokens = word_tokenize(','.join(str(v) for v in corpus_iphone))

# remove stopwords
stopped_iphone_tokens = [word.lower() for word in iphone_tokens if word.lower
                          not in stopword_list]
```

In [218]:
```
freq = FreqDist(stopped_iphone_tokens)
```

In [225]:
```
freq = pd.DataFrame(freq.most_common(25))
```

In [230]:

```
fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=freq.head(10)[0], y=freq.head(10)[1], palette=palette)
ax.set(xlabel="Word",ylabel="Frequency")
plt.ticklabel_format(style='plain',axis='y')
plt.xticks(rotation=70)
plt.title('Most Common Words in Positive Tweets About iPhones')
plt.show()
```



1.11.4 Recommendation

[...]