EDA, Preprocessing, and Tweet Analysis Notebook

In [1]:

```
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=
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, plot_confu
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, Gradie
from sklearn.naive_bayes import BernoulliNB, CategoricalNB, GaussianNB, Multino
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer, T
from sklearn.model_selection import GridSearchCV, train_test_split

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 [2]:
```
nlp = spacy.load('en_core_web_sm')
```

In [3]:
```
print(stopwords)
print(nlp.Defaults.stop_words)
```

```
<WordListCorpusReader in '.../corpora/stopwords' (not loaded yet)>
{'none', 'hereafter', 'we', 'anyhow', 'how', 'such', 'before', 'enough', 'towards', 'do', 'must', 'e
s', 'whole', 'beyond', 'whom', 'however', "'re", 'have', 'too', 'our', 'toward', 'somewhere', 'there
y', 'hereupon', 'until', 'move', 'anything', 'a', 'that', 'used', 'front', 'might', 'nine', 'whence'
m', 'using', 'but', 'yourself', 'therefore', 'often', 'say', 'been', 'various', 'please', 'doing', '
'really', 'four', '‘re', 'someone', 'give', 'several', 'never', 'these', 'only', 'into', 'quite', 'y
re', 'elsewhere', 'besides', 'in', 'whatever', 'became', 'fifteen', 'of', 'latter', 'among', 'anywhe
h', 'call', 'whereupon', 'others', 'hence', 'mostly', 'i', 'your', 'about', 'along', 'becoming', 'ac
'through', 'become', 'get', 'three', 'n‘t', 'than', 'regarding', 'once', 'somehow', 'already', 'ther
r', "'s", 'whereas', 'all', 'their', 'whenever', 'an', 'less', 'the', 'you', 'why', 'together', 'bet
d', 'nowhere', 'eight', 'over', 'another', 'beforehand', 'whither', 'otherwise', 'everyone', 'unless
t', 'then', 'when', 'himself', 'throughout', 'fifty', '‘m', 'wherever', 'below', 'formerly', 'furthe
y', 'indeed', 'last', 'because', 'what', 'becomes', 'one', 'nor', 'after', 'herself', 'off', 'her',
'‘ll', 'amongst', 'even', 'or', 'as', 'either', 'full', 'meanwhile', 'same', 'much', 'under', 'after
'thereby', 'former', 'amount', "'ve", 'out', 'always', 'where', 'something', 'was', 'rather', 'will'
er', 'many', 'could', 'go', 'alone', 'hereby', 'not', '’m', '‘ve', 'twelve', 'by', 'without', 'almos
'to', 'very', 'he', 'everywhere', 'see', 'so', 'she', 'moreover', 'behind', 'eleven', 'just', 'my',
t', 'is', 'next', "'d", '’re', 'five', 'six', 'onto', "'m", 'neither', 'per', 'on', 'sixty', 'within
'n’t', 'nothing', 'anyone', 'its', 'thence', 'and', '’s', 'side', 'third', '‘d', 'would', 'least', '
d', 'can', 'ca', 'sometimes', 'show', 'mine', 'cannot', 'seeming', 'ourselves', 'namely', "n't", 'ea
tself', 'while', 'for', 'has', 'against', 'noone', 'whose', 'yet', 'made', 'although', 'does', 'perh
ost', '‘s', 'myself', 'take', 'from', 'back', 'sometime', 'again', 'ever', 'keep', 'be', 'thru', 'st
ia', 'down', 'wherein', 'seem', 'did', 'his', 'herein', 'other', 'them', 'here', 'twenty', 'whoever'
irst'}
```

In [4]:
```
df = pd.read_csv('data/product_tweets.csv',encoding='latin1')
```

In [5]:
```
df.head()
```

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_dir
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 [6]:
```
df['is_there_an_emotion_directed_at_a_brand_or_product'].unique()
```

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

In [7]:
```
df = df.rename(columns= {'is_there_an_emotion_directed_at_a_brand_or_product'
                         :'Emotion','emotion_in_tweet_is_directed_at': 'Platfo
```

In [8]:
```
df = df.rename(columns= {'tweet_text': 'Tweet'})
```

In [9]:
```
df.head() # want to remove the @'name' in the tweet
```

	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 [10]:
```
df_dummify = pd.get_dummies(df['Emotion'])
```

In [11]:
```
df_dummify.head()
```

	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 [12]:
```
df_dummify.sum() # class bias
```

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

In [13]:
```
df.info()
df = pd.merge(df, df_dummify, how='outer',on=df.index) # ran this code, dummify
```

```
<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 [14]:
```
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 [15]:
```
df.head()
```

	key_0	Tweet	Platform	Emotion	I can't tell	Negative emotion	N
0	0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative emotion	0	1	0
1	1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive emotion	0	0	0
2	2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive emotion	0	0	0
3	3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Negative emotion	0	1	0
4	4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Positive emotion	0	0	0

In [16]:
```
df = df.rename(columns = {"I can't tell": "Uncertain", 'Negative emotion': 'Neg
                          , 'No emotion toward brand or product': 'No Emotion'
                          , 'Positive emotion':'Positive'})
```

In [17]:
```
df = df.drop(columns='key_0')
df.head()
```

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

In [18]:
```
corpus = list(df['Tweet'])
corpus[:10]
```

```
['.@wesley83 I have a 3G iPhone. After 3 hrs tweeting at #RISE_Austin, it was dead!  I need to upgra
 "@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its desi
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 app. #sxsw",
 "@sxtxstate great stuff on Fri #SXSW: Marissa Mayer (Google), Tim O'Reilly (tech books/conferences)
s)",
 '@teachntech00 New iPad Apps For #SpeechTherapy And Communication Are Showcased At The #SXSW Confer
M) #iear #edchat #asd',
 nan,
 '#SXSW is just starting, #CTIA is around the corner and #googleio is only a hop skip and a jump fro
roid fan',
 'Beautifully smart and simple idea RT @madebymany @thenextweb wrote about our #hollergram iPad app
bit.ly/ieaVOB',)
 'Counting down the days to #sxsw plus strong Canadian dollar means stock up on Apple gear']
```

Tokenize the Words

In [19]:
```
tokenz = word_tokenize(','.join(str(v) for v in corpus))
```

In [20]:
```
tokenz[:10]
```

```
['.', '@', 'wesley83', 'I', 'have', 'a', '3G', 'iPhone', '.', 'After']
```

Create Stopwords List

In [21]:
```
stopword_list = list(nlp.Defaults.stop_words)
len(nlp.Defaults.stop_words)
```

```
326
```

In [22]:
```
stopword_list
```

```
 'from',
 'back',
 'sometime',
 'again',
 'ever',
 'keep',
 'be',
 'thru',
 'still',
 'both',
 'nevertheless',
 'via',
 'down',
 'wherein',
 'seem',
 'did',
 'his',
 'herein',
 'other',
 'them',
 'here',
 'twenty',
 'whoever',
 'yours',
 'since',
```

In [23]:
```
stopword_list.extend(string.punctuation)
```

In [24]:
```
len(stopword_list)
```

```
358
```

In [25]:
```
stopword_list.extend(stopwords.words('english'))
```

In [26]:
```
len(stopword_list)
```

```
537
```

In [27]:
```
additional_punc = ['“','”','...',"'",'’','``','https','rt','\.+']
stopword_list.extend(additional_punc)
stopword_list[-10:]
```

```
["wouldn't", '“', '”', '...', "'", '’', '``', 'https', 'rt', '\\.+']
```

Remove stopwords and additional punctuation from the data

In [28]:
```
stopped_tokenz = [word.lower() for word in tokenz if word.lower() not in stopwo
```

In [29]:

```
freq = FreqDist(stopped_tokenz)
freq.most_common(50)
```

```
[('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)]
```

Lemmatize the Data and use Regex to find and remove URL misc

In [30]:
```
additional_misc = ['sxsw','mention',r'[a-zA-Z]+\'?s]',r"(http[s]?://\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]?',
stopword_list.extend(additional_misc)
stopword_list.extend(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'])
```

In [31]:
```
lemmatizer = WordNetLemmatizer()
```

In [32]:
```
clean_stopped_tokenz = [word.lower() for word in stopped_tokenz if word not in
clean_lemmatized_tokenz = [lemmatizer.lemmatize(word.lower()) for word in stopp
```

In [33]:
```
freq_clean_lemma = FreqDist(clean_lemmatized_tokenz)
freq_lemma = freq_clean_lemma.most_common(5000)
freq_lemma2 = freq_clean_lemma.most_common(25)
```

In [34]:
```
total_word_count = len(clean_lemmatized_tokenz)
```

In [35]:
```
lemma_word_count = sum(freq_clean_lemma.values()) # just a number
```

In [36]:
```
for word in freq_lemma2:
    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 [37]:
```
# 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')
```

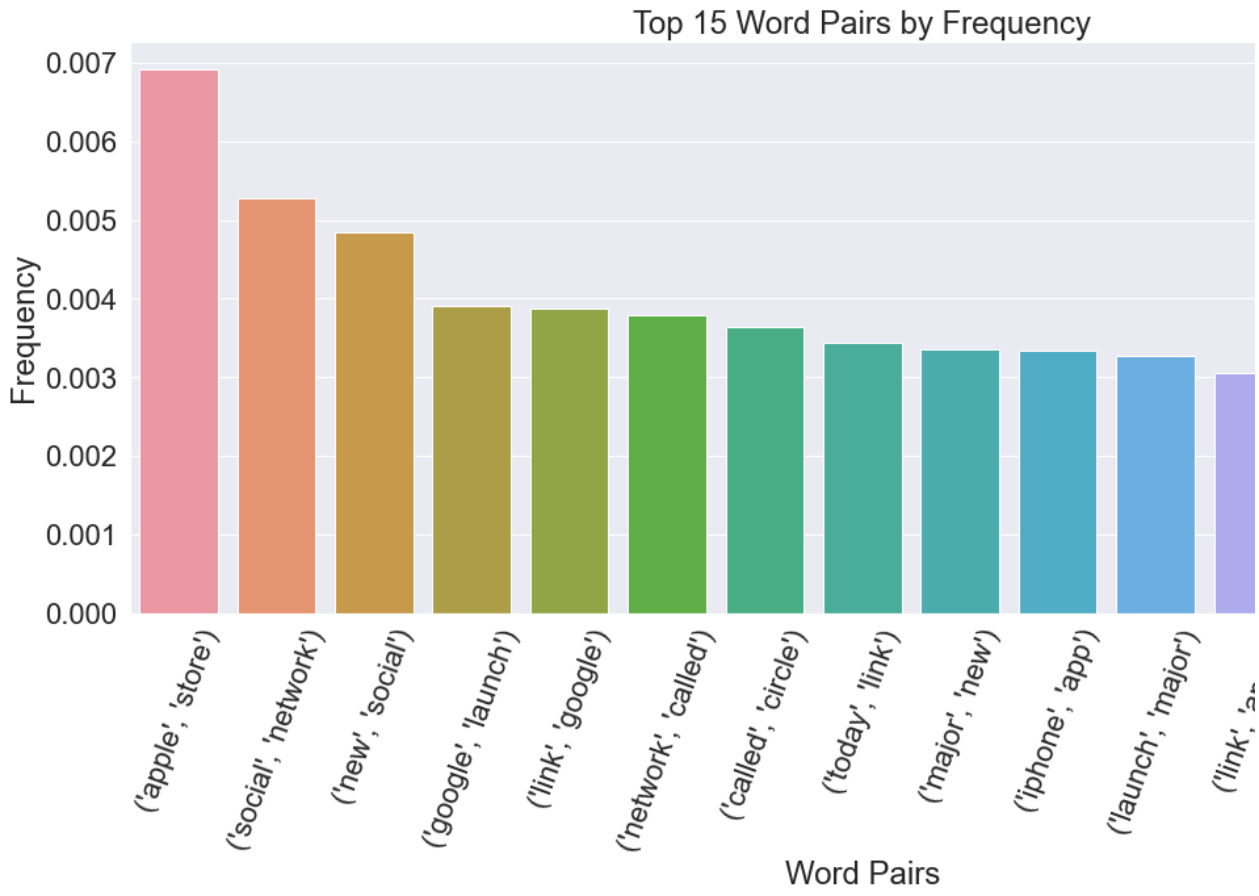
In [38]:
```
bigram_measures = nltk.collocations.BigramAssocMeasures()
tweet_finder = nltk.BigramCollocationFinder.from_words(clean_lemmatized_tokenz)
tweets_scored = tweet_finder.score_ngrams(bigram_measures.raw_freq)
```

In [39]:
```
word_pairs = pd.DataFrame(tweets_scored, columns=["Word","Freq"]).head(20)
word_pairs
```

	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
11	(link, apple)	0.003055
12	(pop-up, store)	0.002870
13	(possibly, today)	0.002731
14	(circle, possibly)	0.002720
15	(apple, opening)	0.002615

In [75]:

```
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'],
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 [41]:

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

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

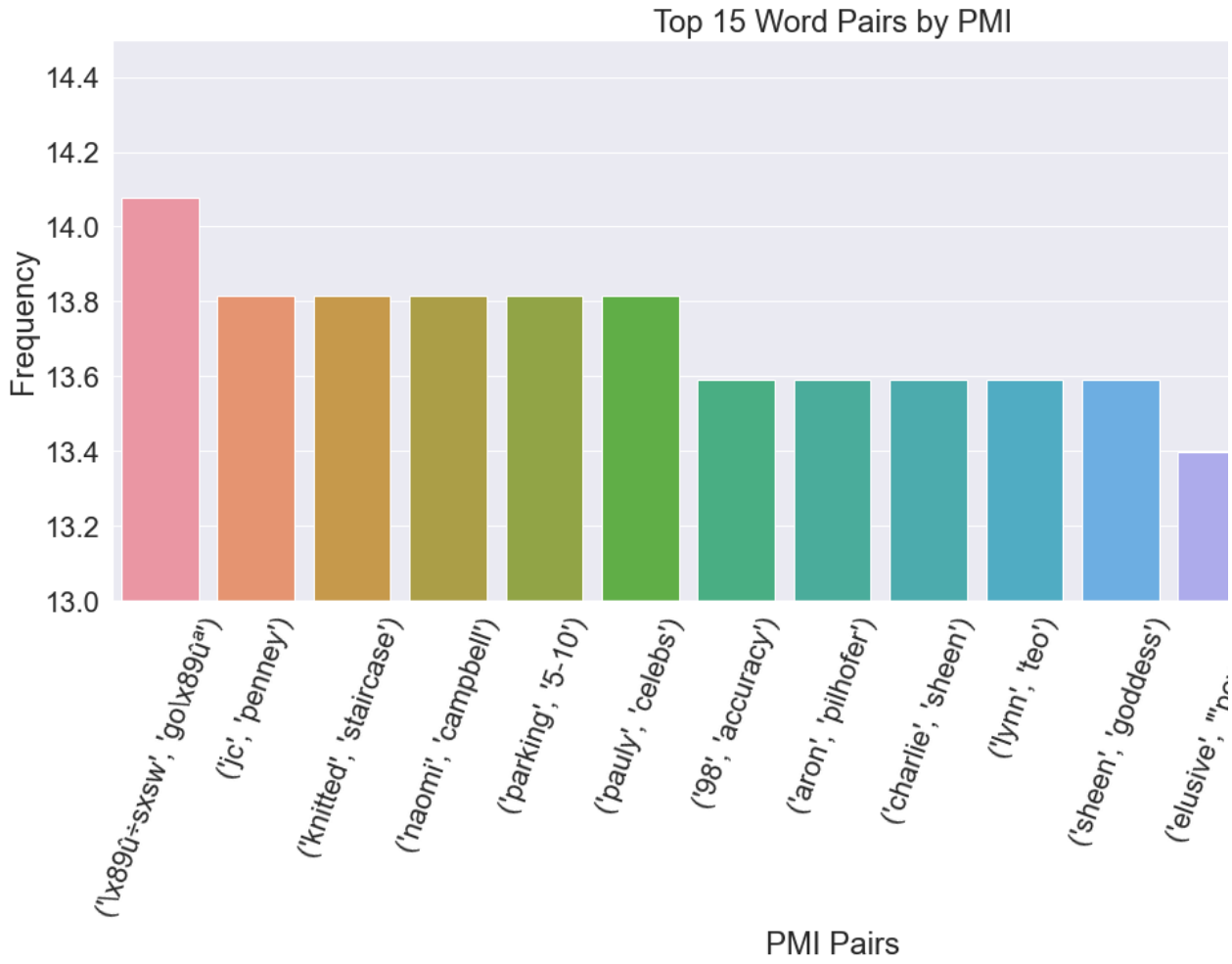
In [42]:

```
PMI_list = pd.DataFrame(tweet_pmi_scored, columns=["Words","PMI"]).head(20)
PMI_list
```

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
12	(zazzlsxsw, you□ûªll)	13.398911
13	(cameron, sinclair)	13.398911
14	(sinclair, spearhead)	13.398911
15	(staircase, attendance)	13.398911
16	(likeability, virgin)	13.328521
17	(14-day, return)	13.228986
18	(launchrock, comp)	13.228986
19	(participating, launchrock)	13.228986

In [43]:

```
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'], pale
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 [44]:

```
df1 = df
df.head()
```

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

In [45]:
```
df1 = df1.drop(columns=['Uncertain','No Emotion'])
# Turn negative and positive columns into one column of just negatives and pos
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()
```

	Tweet	Platform	Emotion	Positive_B
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

Create upsampled data, train and test sets

In [46]:
```
from sklearn.utils import resample
```

In [47]:
```
df_majority = df1.loc[df1['Positive_Bin']==1]
df_minority = df1.loc[df1['Positive_Bin']==0]
```

In [48]:
```
df_minority.shape
```

```
(570, 4)
```

In [49]:
```
df_majority.shape
```

```
(2978, 4)
```

In [50]:
```
df_min_sample = resample(df_minority, replace=True, n_samples=1000, random_stat
```

In [51]:
```
df_maj_sample = resample(df_majority, replace=True, n_samples=2500, random_stat
```

In [52]:
```
df_upsampled = pd.concat([df_min_sample, df_maj_sample], axis=0)
df_upsampled.shape
```

```
(3500, 4)
```

In [53]:
```
X, y = df_upsampled['Tweet'], df_upsampled['Positive_Bin']
```

Train/Test Split

In [54]:
```
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 [55]:
```
scaler_object = MaxAbsScaler()
```

In [56]:
```
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 [57]:
```
y_train.value_counts(0)
y_test.value_counts(1)
```

```
2020-12-17 13:23:40,433 : INFO : NumExpr defaulting to 8 threads.
```

```
1    0.683429
0    0.316571
Name: Positive_Bin, dtype: float64
```

Vectorize, Lemmatize with Count Vectorizer and Tf Idf

In [58]:
```
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 [59]:
```
# for row in X_train:
#     for word in row:
#         lemmatizer.lemmatize(X_train[row][word])
# return X_train[word][row]
```

In [60]:
```
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
with your preprocessing. Tokenizing the stop words generated tokens [":'[", ':/', 'a-z', 'a-za-z', '
p_words.
  warnings.warn('Your stop_words may be inconsistent with '
```

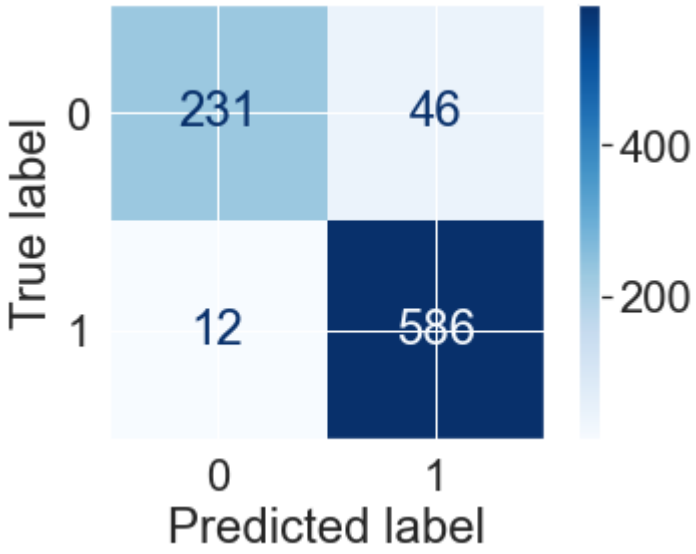
In [61]:
```
ran_for = RandomForestClassifier(class_weight='balanced')
model = ran_for.fit(X_train_count, y_train)
```

In [62]:
```
y_hat_test = model.predict(X_test_count)
```

Evaluate Models

In [63]:
```
evaluate_model(y_test, y_hat_test, X_test_count,clf=model) # 1 denotes Positive
```

	precision	recall	f1-score	support
0	0.95	0.83	0.89	277
1	0.93	0.98	0.95	598
accuracy			0.93	875
macro avg	0.94	0.91	0.92	875
weighted avg	0.93	0.93	0.93	875



In [64]:
```
tf_idf_vectorizer = TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                    stop_words=stopword_list,decode_error='igno
```

In [65]:
```
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
with your preprocessing. Tokenizing the stop words generated tokens [":'[", ':/', 'a-z', 'a-za-z', '
p_words.
  warnings.warn('Your stop_words may be inconsistent with '


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

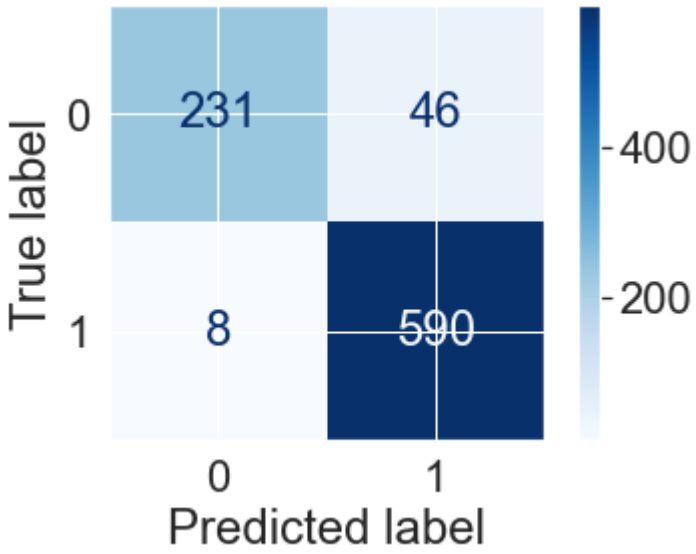
In [66]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [67]:
```
ran_for = RandomForestClassifier(class_weight='balanced')
model_tf_idf = ran_for.fit(X_train_tf_idf,y_train)
```

In [68]:
```
y_hat_tf_idf = model_tf_idf.predict(X_test_count)
```

In [69]:
```
evaluate_model(y_test, y_hat_tf_idf, X_test_tf_idf,clf=model_tf_idf) # slightly
```

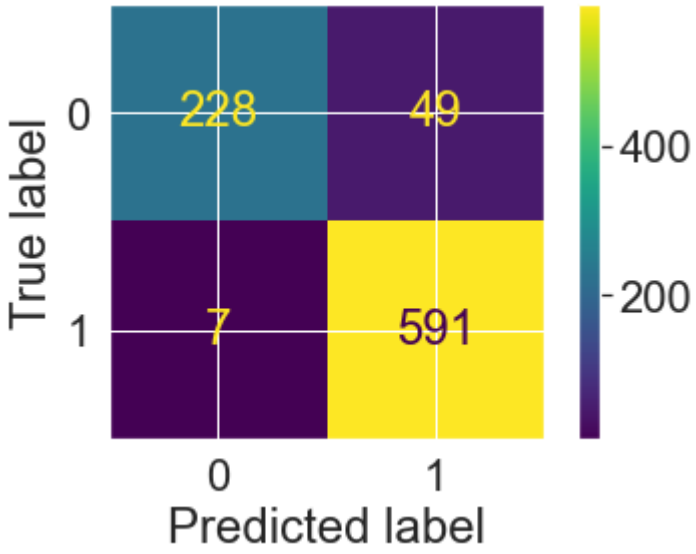
	precision	recall	f1-score	support
0	0.87	0.64	0.74	277
1	0.85	0.96	0.90	598
accuracy			0.86	875
macro avg	0.86	0.80	0.82	875
weighted avg	0.86	0.86	0.85	875



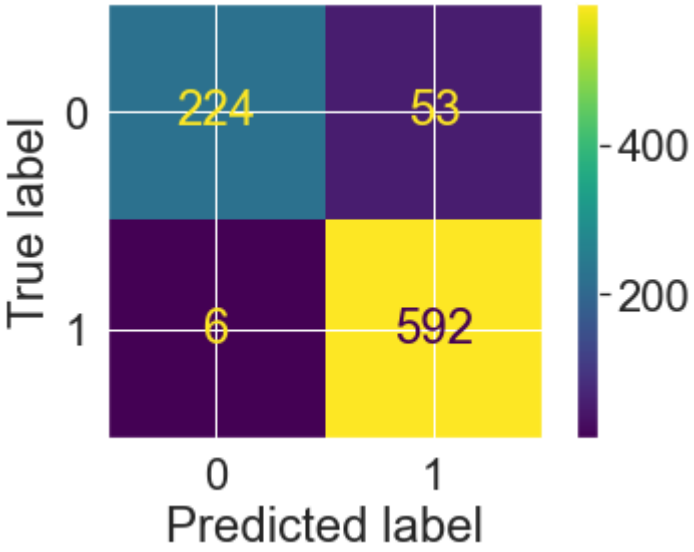
In [70]:

```
ran_for = RandomForestClassifier()
ada_clf = AdaBoostClassifier()
gb_clf = GradientBoostingClassifier()

models = [ran_for, ada_clf, gb_clf]

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

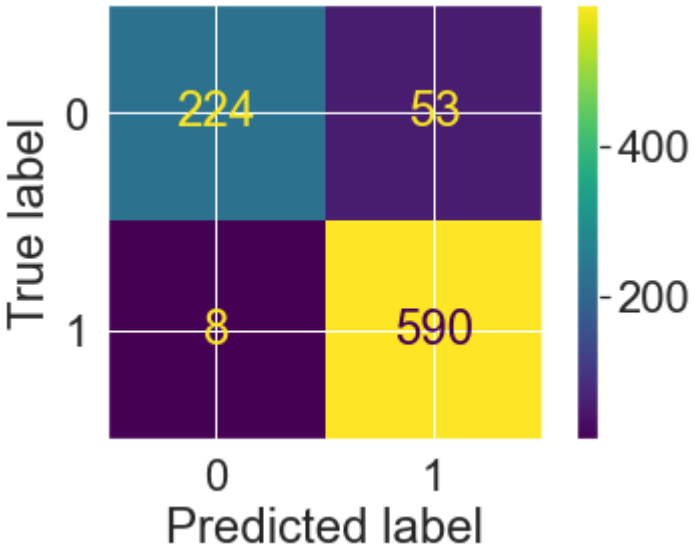
```
Accuracy Score:  0.936
Precision Score:  0.9234375
Recall Score:  0.9882943143812709
F1 Score:  0.9547657512116317
RandomForestClassifier()   0.936
```




```
Accuracy Score:  0.9325714285714286
Precision Score:  0.9178294573643411
Recall Score:  0.9899665551839465
F1 Score:  0.9525341914722445
RandomForestClassifier()   0.9325714285714286
```

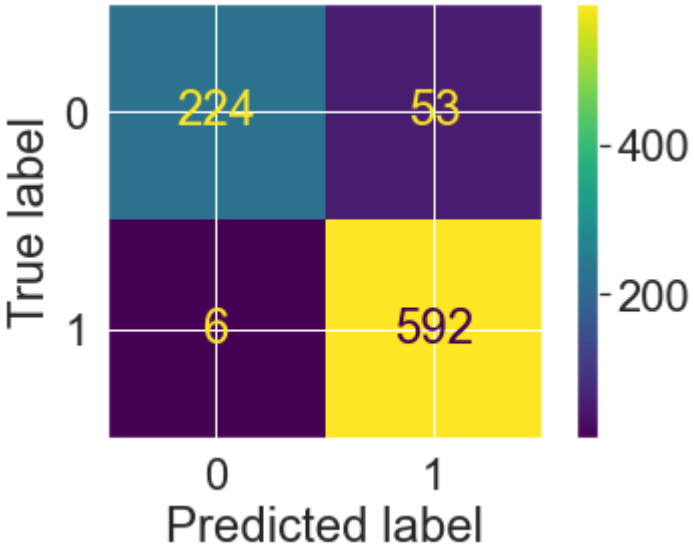



```
Accuracy Score:  0.9302857142857143
Precision Score:  0.9175738724727839
Recall Score:  0.9866220735785953
F1 Score:  0.9508460918614021
RandomForestClassifier()   0.9302857142857143
```

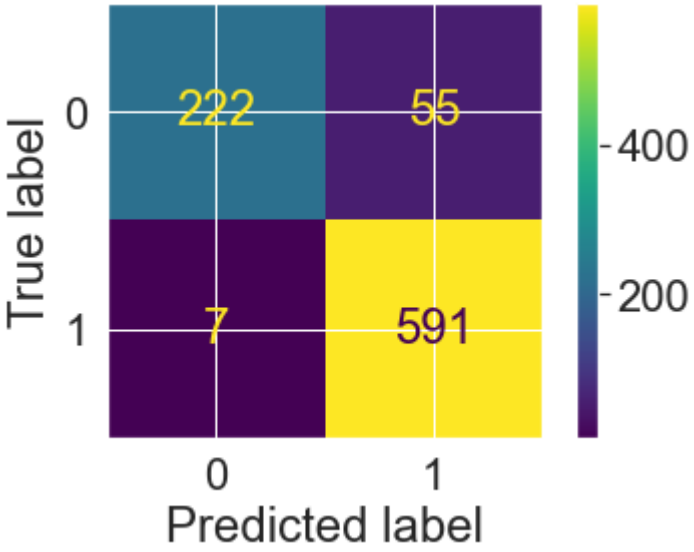


In [71]:
```
for model in models:
    single_model_opt(ran_for, X_train_tf_idf, y_train, X_test_tf_idf, y_test)
```

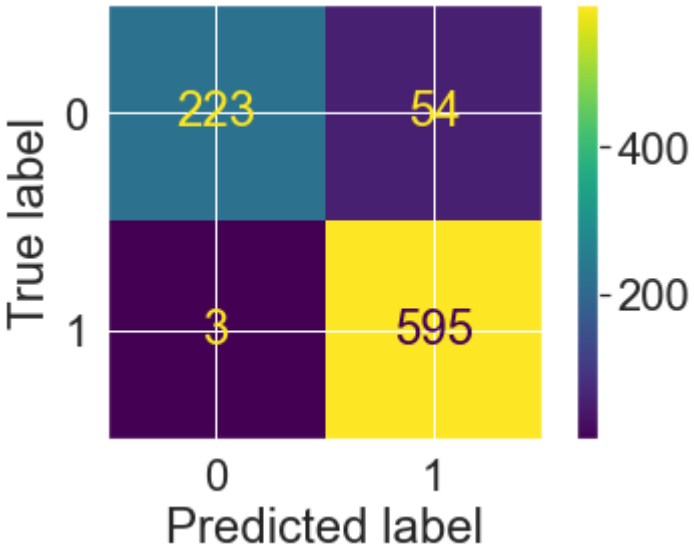
```
Accuracy Score:  0.9325714285714286
Precision Score:  0.9178294573643411
Recall Score:  0.9899665551839465
F1 Score:  0.9525341914722445
RandomForestClassifier()   0.9325714285714286
```




```
Accuracy Score:  0.9291428571428572
Precision Score:  0.9148606811145511
Recall Score:  0.9882943143812709
F1 Score:  0.95016077170418
RandomForestClassifier()   0.9291428571428572
```




```
Accuracy Score:  0.9348571428571428
Precision Score:  0.9167950693374423
Recall Score:  0.9949832775919732
F1 Score:  0.9542902967121091
RandomForestClassifier()   0.9348571428571428
```



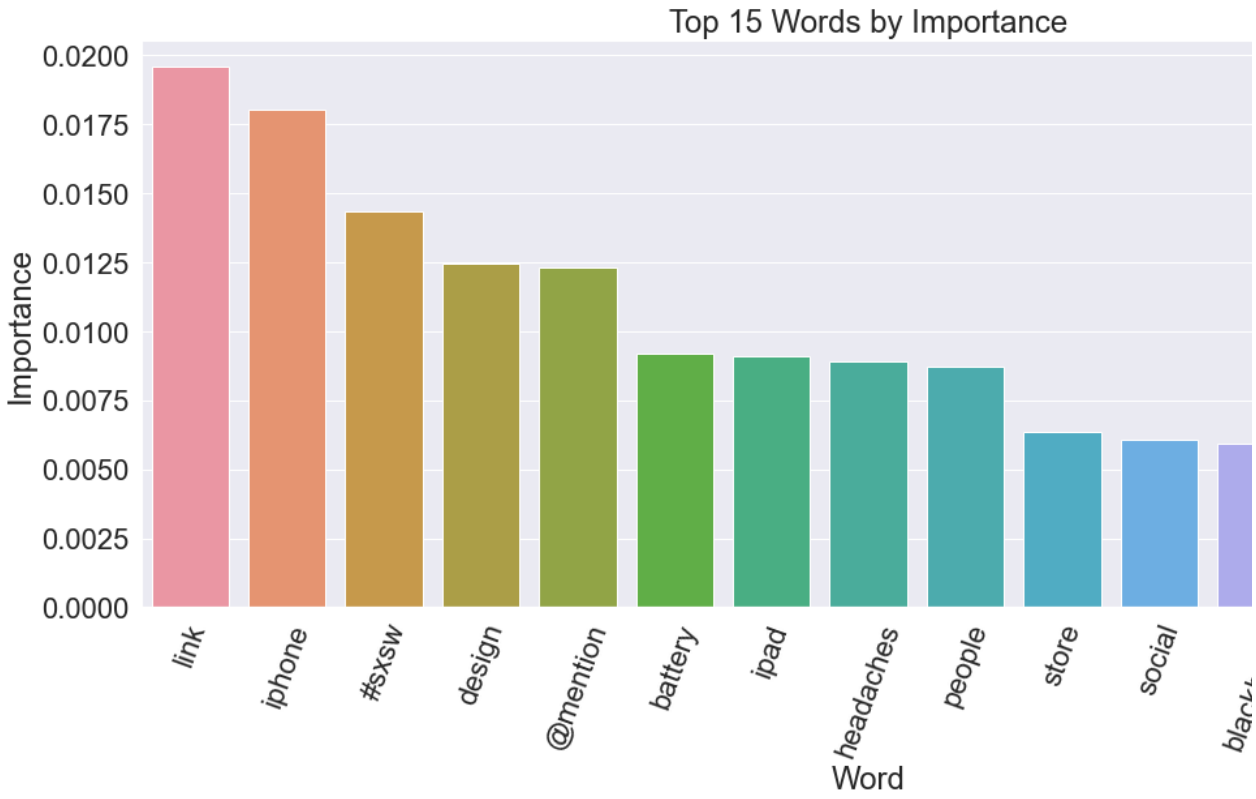
In [72]:
```
tf_idf_vectorizer.get_feature_names()
```

```
 '#behance',
 '#bestappever',
 '#betainvites',
 '#bettercloud',
 '#bettersearch',
 '#betterthingstodo',
 '#beyondwc',
 '#bing',
 '#bizzy',
 '#blackberry',
 '#boom',
 '#booyah',
 '#brainwashed',
 '#brian_lam',
 '#brk',
 '#broadcastr',
 '#browserwars',
 '#cartoon',
 '#catphysics',
 '#cbatsxsw',
 '#ces',
 '#channels',
 '#chargin2diffphonesatonce',
 '#checkins',
 '#circles'
```

In [73]:
```
importance = pd.Series(ran_for.feature_importances_,index=tf_idf_vectorizer.get
importance = pd.DataFrame(importance).sort_values(by=0,ascending=False)
```

In [80]:

```
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
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()
```



In [81]:

```
vectorizer = CountVectorizer()
tf_transform = TfidfTransformer(use_idf=True)
```

In [82]:
```
text_pipe = Pipeline(steps=[
    ('count_vectorizer',vectorizer),
    ('tf_transformer',tf_transform)])
```

In [83]:
```
RandomForestClassifier(class_weight='balanced')
```

```
RandomForestClassifier(class_weight='balanced')
```

In [84]:
```
full_pipe = Pipeline(steps=[
    ('text_pipe',text_pipe),
    ('clf',RandomForestClassifier(class_weight='balanced'))
])
```

In [85]:
```
X_train_pipe = text_pipe.fit_transform(X_train)
```

In [86]:
```
X_test_pipe = text_pipe.transform(X_test)
```

In [87]:
```
X_train_pipe
```

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

In [88]:
```
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 [89]:

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

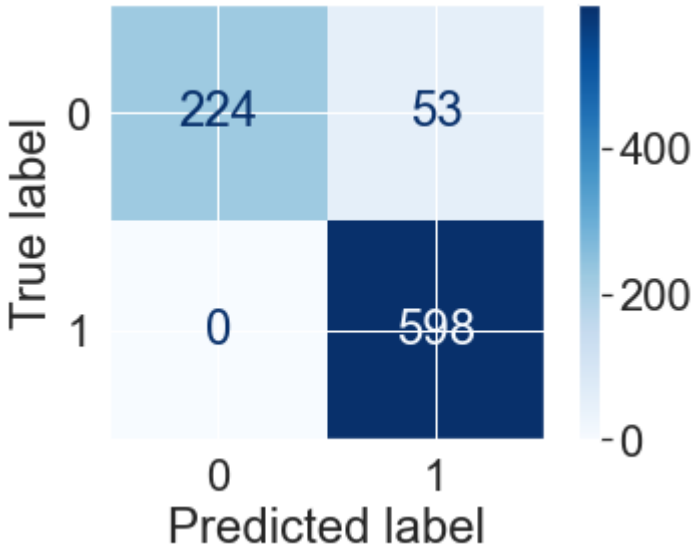
```
  warnings.warn('Your stop_words may be inconsistent with '
C:\Users\josep\anaconda3\lib\site-packages\sklearn\feature_extraction\text.py:383: UserWarning: Your
with your preprocessing. Tokenizing the stop words generated tokens ['http'] not in stop_words.
  warnings.warn('Your stop_words may be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

In [90]:

```
best_pipe = grid.best_estimator_
y_hat_test = grid.predict(X_test)
```

In [91]:
```
evaluate_model(y_test,y_hat_test,X_test,best_pipe)
```

```
              precision    recall  f1-score   support

           0       1.00      0.81      0.89       277
           1       0.92      1.00      0.96       598

    accuracy                           0.94       875
   macro avg       0.96      0.90      0.93       875
weighted avg       0.94      0.94      0.94       875
```



In [92]:
```
X_train_pipe.shape
```

```
(2625, 4256)
```

In [93]:
```
features = text_pipe.named_steps['count_vectorizer'].get_feature_names()
features[:10]
```

```
['000', '02', '03', '0310apple', '08', '10', '100', '100s', '101', '106']
```

In [94]:
```
bigram_measures = nltk.collocations.BigramAssocMeasures()
tweet_finder = nltk.BigramCollocationFinder.from_words(clean_lemmatized_tokenz)
tweets_scored = tweet_finder.score_ngrams(bigram_measures.raw_freq)
```

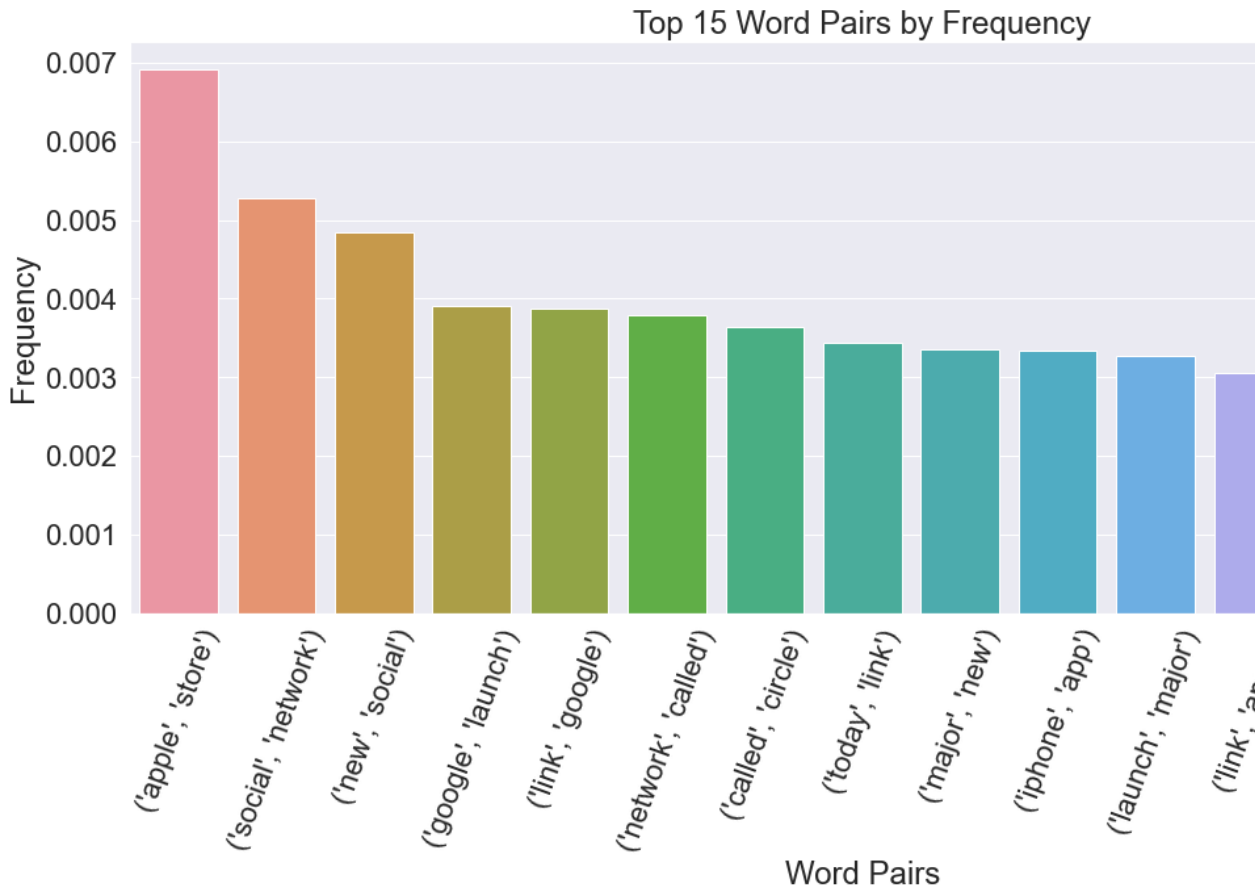
In [95]:
```
bigram1 = pd.DataFrame(tweets_scored, columns=['Words','Freq'])
bigram1
```

	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
...	...	...
42702	(åç, complete)	0.000012
42703	(åçwhat, tech)	0.000012
42704	(åè, android)	0.000012
42705	(åè, ubersoc)	0.000012
42706	(ìù□±g, wish)	0.000012

42707 rows × 2 columns

In [96]:

```
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'], palet
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()
```



Deep NLP using Keras NN

In [97]:

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import models, layers, optimizers
```

In [98]:
```
model = 0
```

In [99]:
```
y = np.asarray(df_upsampled['Positive_Bin']).astype('float32').reshape((-1,1))
X = np.asarray(df_upsampled['Tweet'])
```

In [100]:
```
X.dtype
```

```
dtype('O')
```

In [101]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(list(X))
sequences = tokenizer.texts_to_sequences(X)
X = sequence.pad_sequences(sequences,maxlen=100)
```

In [102]:
```
tokenizer.word_counts
```

```
('fast', 42),
('among', 23),
('digital', 37),
('delegates', 19),
('rule', 2),
('no', 161),
('more', 102),
('ooing', 2),
('and', 636),
('ahing', 2),
('over', 68),
('your', 168),
('we', 86),
('get', 199),
('it', 480),
('its', 58),
('big', 34),
('deal', 6),
('everybody', 3),
('has', 138),
('one', 149),
('overheard', 9),
('interactive', 34),
('quot', 667),
```

In [103]:
```
vocab_size = len(tokenizer.word_counts)
seq_len = X.shape[1]
```

[In [104]]:
```
vocab_size
seq_len
```

```
100
```

[In [105]]:
```
print(type(X),X.shape)
print(type(y),y.shape)
```

```
<class 'numpy.ndarray'> (3500, 100)
<class 'numpy.ndarray'> (3500, 1)
```

[In [106]]:
```
X = np.asarray(X).astype('float32')
```

[In [107]]:
```
print(type(X),X.shape)
print(type(y),y.shape)
```

```
<class 'numpy.ndarray'> (3500, 100)
<class 'numpy.ndarray'> (3500, 1)
```

[In [108]]:
```
# X_train, y_train, X_test, y_test = train_test_split(X, y, random_state=42, te
```

[In [109]]:
```
def create_model(vocab_size,seq_len):

    model = Sequential()
    embedding_size = 128
    model.add(Embedding(vocab_size, seq_len, input_length=seq_len))
    model.add(Dense(16,input_dim=2, activation='relu'))
    model.add(LSTM(8,input_dim=2, activation='relu'))
    model.add(Dense(2, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['precision'])

    model.summary()

    return model
```

In [110]:
```
model = create_model(vocab_size,seq_len)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 100)          481600
_________________________________________________________________
dense (Dense)                (None, 100, 16)           1616
_________________________________________________________________
lstm (LSTM)                  (None, 8)                 800
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 18
=================================================================
Total params: 484,034
Trainable params: 484,034
Non-trainable params: 0
_________________________________________________________________
```

[n [111]:
```
model.fit(X,y, batch_size=32, epochs=5, verbose=1, validation_split=.2)
```

```
Epoch 1/5


---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-111-c1e6834d6817> in <module>
----> 1 model.fit(X,y, batch_size=32, epochs=5, verbose=1, validation_split=.2)

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\training.py in fit(se
rbose, callbacks, validation_split, validation_data, shuffle, class_weight, sample_weight, initial_e
n_steps, validation_batch_size, validation_freq, max_queue_size, workers, use_multiprocessing)
   1098                 _r=1):
   1099               callbacks.on_train_batch_begin(step)
-> 1100               tmp_logs = self.train_function(iterator)
   1101               if data_handler.should_sync:
   1102                 context.async_wait()

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\def_function.py in __call__(
    826     tracing_count = self.experimental_get_tracing_count()
    827     with trace.Trace(self._name) as tm:
--> 828       result = self._call(*args, **kwds)
    829       compiler = "xla" if self._experimental_compile else "nonXla"
    830       new_tracing_count = self.experimental_get_tracing_count()

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\def_function.py in _call(sel
    869       # This is the first call of __call__, so we have to initialize.
    870       initializers = []
--> 871       self._initialize(args, kwds, add_initializers_to=initializers)
    872     finally:
    873       # At this point we know that the initialization is complete (or less

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\def_function.py in _initiali
izers_to)
    723     self._graph_deleter = FunctionDeleter(self._lifted_initializer_graph)
    724     self._concrete_stateful_fn = (
--> 725         self._stateful_fn._get_concrete_function_internal_garbage_collected(  # pylint: disa
    726             *args, **kwds))
    727

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\function.py in _get_concrete
cted(self, *args, **kwargs)
   2967       args, kwargs = None, None
   2968     with self._lock:
-> 2969       graph_function, _ = self._maybe_define_function(args, kwargs)
   2970     return graph_function
   2971

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\function.py in _maybe_define
   3359
   3360           self._function_cache.missed.add(call_context_key)
-> 3361           graph_function = self._create_graph_function(args, kwargs)
   3362           self._function_cache.primary[cache_key] = graph_function
   3363
```

```
~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\function.py in _create_graph
erride_flat_arg_shapes)
   3194     arg_names = base_arg_names + missing_arg_names
   3195     graph_function = ConcreteFunction(
-> 3196         func_graph_module.func_graph_from_py_func(
   3197             self._name,
   3198             self._python_function,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\framework\func_graph.py in func_gr
nc, args, kwargs, signature, func_graph, autograph, autograph_options, add_control_dependencies, arg
ions, capture_by_value, override_flat_arg_shapes)
    988         _, original_func = tf_decorator.unwrap(python_func)
    989
--> 990       func_outputs = python_func(*func_args, **func_kwargs)
    991
    992       # invariant: `func_outputs` contains only Tensors, CompositeTensors,

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\def_function.py in wrapped_f
    632             xla_context.Exit()
    633         else:
--> 634           out = weak_wrapped_fn().__wrapped__(*args, **kwds)
    635         return out
    636

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\framework\func_graph.py in wrapper
    975           except Exception as e:  # pylint:disable=broad-except
    976             if hasattr(e, "ag_error_metadata"):
--> 977               raise e.ag_error_metadata.to_exception(e)
    978             else:
    979               raise

ValueError: in user code:

    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\trai
        return step_function(self, iterator)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\trai
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\distribute\distri
        return self._extended.call_for_each_replica(fn, args=args, kwargs=kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\distribute\distri
eplica
        return self._call_for_each_replica(fn, args, kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\distribute\distri
replica
        return fn(*args, **kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\trai
        outputs = model.train_step(data)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\trai
        loss = self.compiled_loss(
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\comp
        loss_value = loss_obj(y_t, y_p, sample_weight=sw)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\losses.py:1
        losses = call_fn(y_true, y_pred)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\losses.py:2
        return ag_fn(y_true, y_pred, **self._fn_kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\dispatch.py:
        return target(*args, **kwargs)
```

```
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\losses.py:1
        K.binary_crossentropy(y_true, y_pred, from_logits=from_logits), axis=-1)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\dispatch.py:
        return target(*args, **kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\backend.py:
        return nn.sigmoid_cross_entropy_with_logits(labels=target, logits=output)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\dispatch.py:
        return target(*args, **kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\nn_impl.py:17
its
        raise ValueError("logits and labels must have the same shape (%s vs %s)" %

    ValueError: logits and labels must have the same shape ((None, 2) vs (None, 1))
```

In [ ]:

Deep NLP using Word2Vec NN

In [112]:
```
from nltk import word_tokenize
```

In [113]:
```
data = df_upsampled['Tweet'].map(word_tokenize)
```

In [114]:
```
data[:10]
```

```
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
```

[n [115]:
```
model_W2V = Word2Vec(data, size =100, window=5, min_count=1, workers=4)
```

```
2020-12-17 13:40:40,106 : INFO : collecting all words and their counts
2020-12-17 13:40:40,108 : INFO : PROGRESS: at sentence #0, processed 0 words, keeping 0 word types
2020-12-17 13:40:40,124 : INFO : collected 5920 word types from a corpus of 86715 raw words and 3500
2020-12-17 13:40:40,124 : INFO : Loading a fresh vocabulary
2020-12-17 13:40:40,136 : INFO : effective_min_count=1 retains 5920 unique words (100% of original 5
2020-12-17 13:40:40,137 : INFO : effective_min_count=1 leaves 86715 word corpus (100% of original 86
2020-12-17 13:40:40,155 : INFO : deleting the raw counts dictionary of 5920 items
2020-12-17 13:40:40,156 : INFO : sample=0.001 downsamples 52 most-common words
2020-12-17 13:40:40,157 : INFO : downsampling leaves estimated 56808 word corpus (65.5% of prior 867
2020-12-17 13:40:40,170 : INFO : estimated required memory for 5920 words and 100 dimensions: 769600
2020-12-17 13:40:40,170 : INFO : resetting layer weights
2020-12-17 13:40:41,237 : INFO : training model with 4 workers on 5920 vocabulary and 100 features,
ative=5 window=5
2020-12-17 13:40:41,277 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,281 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,282 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,283 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,284 : INFO : EPOCH - 1 : training on 86715 raw words (56754 effective words) too
s
2020-12-17 13:40:41,327 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,334 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,335 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,336 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,336 : INFO : EPOCH - 2 : training on 86715 raw words (56756 effective words) too
s
2020-12-17 13:40:41,374 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,376 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,379 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,383 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,383 : INFO : EPOCH - 3 : training on 86715 raw words (56740 effective words) too
s
2020-12-17 13:40:41,422 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,423 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,429 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,430 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,430 : INFO : EPOCH - 4 : training on 86715 raw words (56927 effective words) too
s
2020-12-17 13:40:41,468 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,473 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,474 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,475 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,476 : INFO : EPOCH - 5 : training on 86715 raw words (56738 effective words) too
s
2020-12-17 13:40:41,476 : INFO : training on a 433575 raw words (283915 effective words) took 0.2s,
```

[n [116]:
```
model_W2V.train(data,total_examples=model_W2V.corpus_count, epochs=10)
```

```
2020-12-17 13:40:41,481 : WARNING : Effective 'alpha' higher than previous training cycles
2020-12-17 13:40:41,482 : INFO : training model with 4 workers on 5920 vocabulary and 100 features,
ative=5 window=5
2020-12-17 13:40:41,526 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,534 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,535 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,536 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,537 : INFO : EPOCH - 1 : training on 86715 raw words (56869 effective words) too
s
2020-12-17 13:40:41,580 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,582 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,585 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,586 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,587 : INFO : EPOCH - 2 : training on 86715 raw words (56795 effective words) too
s
2020-12-17 13:40:41,628 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,635 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,636 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,637 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,637 : INFO : EPOCH - 3 : training on 86715 raw words (56726 effective words) too
s
2020-12-17 13:40:41,676 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,679 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,682 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,684 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,685 : INFO : EPOCH - 4 : training on 86715 raw words (56869 effective words) too
s
2020-12-17 13:40:41,723 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,730 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,731 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,732 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,733 : INFO : EPOCH - 5 : training on 86715 raw words (56706 effective words) too
s
2020-12-17 13:40:41,773 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,780 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,781 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,782 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,783 : INFO : EPOCH - 6 : training on 86715 raw words (56870 effective words) too
s
2020-12-17 13:40:41,823 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,824 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,829 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,830 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,830 : INFO : EPOCH - 7 : training on 86715 raw words (56818 effective words) too
s
2020-12-17 13:40:41,869 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,876 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,877 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,877 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,878 : INFO : EPOCH - 8 : training on 86715 raw words (56782 effective words) too
s
2020-12-17 13:40:41,919 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,924 : INFO : worker thread finished; awaiting finish of 2 more threads
```

```
2020-12-17 13:40:41,928 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,931 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,931 : INFO : EPOCH - 9 : training on 86715 raw words (56791 effective words) too
s
2020-12-17 13:40:41,967 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:40:41,973 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:40:41,976 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:40:41,977 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:40:41,978 : INFO : EPOCH - 10 : training on 86715 raw words (56785 effective words) to
s/s
2020-12-17 13:40:41,978 : INFO : training on a 867150 raw words (568011 effective words) took 0.5s,


(568011, 867150)
```

In [117]:

```
wv = model_W2V.wv
```

In [118]:

```
wv.most_similar(positive='good')
```

```
2020-12-17 13:40:41,994 : INFO : precomputing L2-norms of word weight vectors


[('ubiquitous', 0.9811376929283142),
 ('vCards', 0.9785053133964539),
 ('scan', 0.9746865034103394),
 ('stock', 0.9707913994789124),
 ('Cedar', 0.9658127427101135),
 ('expert', 0.9646450877189636),
 ('kidding', 0.9635860919952393),
 ('festival', 0.9624373912811279),
 ('who', 0.9604045152664185),
 ('They', 0.96031254529953)]
```

[n [119]:
```
wv['help']
```

```
array([-0.27211186,  0.12035151, -0.12265135, -0.11108789,  0.18630037,
       -0.14107627, -0.07484463, -0.0505074 , -0.05206762, -0.1134989 ,
        0.12973954,  0.14086726, -0.16788225, -0.195921  ,  0.34800178,
       -0.22925189, -0.2591963 ,  0.07877848, -0.03581034,  0.16063832,
       -0.03543165, -0.23052013,  0.24819116,  0.14521773,  0.03406265,
       -0.08269138, -0.04750591, -0.06089828,  0.08206239,  0.09894749,
        0.37331358,  0.11721657,  0.08336468,  0.06182181, -0.07090339,
        0.22447316, -0.0353047 , -0.08087381, -0.17277719, -0.1720985 ,
        0.12774001,  0.0384092 ,  0.08139827,  0.21278886,  0.08514681,
       -0.14903894, -0.17091446,  0.1993256 ,  0.0262439 ,  0.17968616,
        0.1645444 , -0.08270817, -0.2259742 ,  0.16819794,  0.18885957,
        0.01270939, -0.3248831 , -0.00892121,  0.13323529,  0.04797692,
        0.0865474 , -0.05909903, -0.00241985, -0.27663928, -0.15219055,
       -0.17028974, -0.2103805 ,  0.07531356, -0.22844034,  0.22754721,
        0.2671476 ,  0.05093441, -0.18557051, -0.05916321,  0.311698  ,
        0.02236485,  0.0262726 , -0.25936982,  0.33196375,  0.02997743,
        0.15310277,  0.33875993, -0.04441934, -0.02704216,  0.31035623,
        0.44584247,  0.16157588, -0.34428036, -0.04670548,  0.2026838 ,
       -0.4937711 ,  0.21007192,  0.24622846,  0.09940264,  0.18856491,
       -0.03996108, -0.19012344, -0.15904604,  0.00843009,  0.3810272 ],
      dtype=float32)
```

[n [120]:
```
wv.vectors
```

```
array([[-9.2346621e-01,  3.5479468e-01, -4.3395674e-01, ...,
         1.4169984e-01,  4.7871822e-01,  1.1071440e+00],
       [-5.8863676e-01,  1.6926926e-01, -4.2991567e-01, ...,
        -3.4050202e-01, -4.6820775e-01,  8.1356061e-01],
       [-3.1976596e-01,  5.9318107e-01,  6.7437464e-01, ...,
         3.3112150e-01, -1.1331043e+00, -1.3226213e-01],
       ...,
       [-4.5420166e-02,  2.6782187e-02, -7.7832495e-03, ...,
        -1.0949606e-02, -5.8055632e-03,  5.2558500e-02],
       [-3.8206968e-03, -2.3486570e-02, -3.4674748e-03, ...,
         4.1763674e-04,  1.3604324e-02, -9.2644654e-03],
       [-1.0814171e-02,  8.1993244e-04, -7.4849804e-03, ...,
        -3.0433828e-02, -2.9406650e-02,  3.7506867e-02]], dtype=float32)
```

In [121]:
```
wv.most_similar(positive=['apple','google'], negative = ['man'])
```

```
[('comes', 0.7372466921806335),
 ('non-Mac/iPad', 0.7325534820556641),
 ('cool', 0.7183238863945007),
 ('technology', 0.6977443695068359),
 ('Except', 0.6904727220535278),
 ('geekdate', 0.6572698950767517),
 ('w/people', 0.6488733291625977),
 ('heating', 0.6453533172607422),
 ('Sigh', 0.6086333394050598),
 ('tmsxsw', 0.6080430150032043)]
```

In [ ]: