EDA, Preprocessing, and Tweet Analysis Notebook

[177]:
```
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=log
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_confusi
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, Gradient
from sklearn.naive_bayes import BernoulliNB, CategoricalNB, GaussianNB, Multinomi
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer, Tfic
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!
```

[178]:
```
nlp = spacy.load('en_core_web_sm')
```

[179]:
```
print(stopwords)
print(nlp.Defaults.stop_words)
```

```
<WordListCorpusReader in 'C:\\Users\\josep\\AppData\\Roaming\\nltk_data\\corpora\\stopwords'>
{'mine', 'seemed', 'n’t', 'via', 'up', 'own', 'when', 'hers', 'but', 'elsewhere', 'yet', '’ve', 'these'
e', 'without', 'be', 'every', 'toward', 'anyway', 'with', 'each', 'upon', 'besides', 'since', 'him', 'm
g', 'she', 'among', 'while', 'n‘t', 'bottom', 'in', 'together', 'last', 'amount', 'none', 'empty', 'may
ing', 'well', 'indeed', 'why', 'yourself', '’re', 'no', 'they', 'back', 'almost', 'any', 'put', 'whom',
s', 'being', 'of', 'who', 'twenty', 'thereafter', 'please', 'thereby', 'see', 'whereby', '‘ve', 'thru',
n', 'unless', 'where', 'however', 'various', 'six', 'something', 'nine', 'move', 'becoming', 'his', 'ou
'i', 'on', 'its', 'thus', 'someone', 'therefore', '‘re', 'otherwise', 'same', 'here', 'seem', 'themselv
e', 'yourselves', '’s', 'very', 'out', 'many', 'my', 'those', 'were', 'used', 'myself', "'ll", 'third',
t', 'go', 'through', 'except', 'other', 'and', 'amongst', 'few', 'are', 'one', 'show', 'thereupon', 'wo
f', 'a', 'can', 'anywhere', 'down', 'whereupon', 'say', '‘d', 'neither', 'becomes', 'your', 'been', '‘m
'there', 'then', 'per', 'throughout', 'wherein', 'their', 'that', 'to', 'everything', 'somewhere', 'aft
'at', 'still', 'formerly', 'himself', 'which', 'whenever', 'across', 'least', 'whereafter', 'often', 's
r', 'anyone', 'thence', 'whence', 'moreover', 'enough', 'noone', 'along', 'eleven', 'into', 'from', 'wh
s', 'former', 'does', 'wherever', 'he', 'give', 'us', 'beside', 'herself', 'sometime', 'cannot', 'befor
s', 'might', 'towards', 'most', 'done', 'over', '’ll', 'hereby', 'under', 'am', 'if', 'onto', "n't", 'f
ing', 'call', 'name', 'rather', '‘s', 'less', 'nobody', 'do', 'nor', 'could', 'latterly', '’m', 'more',
n', 'never', 'ten', 'latter', 'off', 'an', 'ever', 'mostly', 'what', 'hundred', 'further', 'due', 'behi
'too', 'her', 'using', 'namely', 'as', 'several', 'had', 'was', 'did', 'get', 'else', 'than', 'yours',
ecause', 'perhaps', 'side', 'nothing', 'therein', 'sixty', 'somehow', "'d", 'whoever', 'once', 'them',
e', 'until', 'three', 'about', 'must', 'anyhow', 'such', 'already', 'above', 'everyone', 'eight', 'has'
ys', 'have', 'herein', 'will', 'ca', 'seems', 'four', 'another', 'only', 'this', 'below', 'become', 'me
d', 'part', 'by', 'hereafter', 'twelve', 'nowhere', 'seeming', 'hereupon', 'against', 'full', 'much', '
'or'}
```

[180]:
```
df = pd.read_csv('data/product_tweets.csv',encoding='latin1')
```

[181]:
```
df.head()
```

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_direct
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

[182]:
```
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)
```

[183]:
```
df = df.rename(columns= {'is_there_an_emotion_directed_at_a_brand_or_product'
                         :'Emotion','emotion_in_tweet_is_directed_at': 'Platform'
```

[184]:
```
df = df.rename(columns= {'tweet_text': 'Tweet'})
```

[185]:
```
df.head()
```

	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

[186]:
```
df_dummify = pd.get_dummies(df['Emotion'])
```

[187]:
```
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

[188]:
```
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
```

[189]:
```
df.info()
df = pd.merge(df, df_dummify, how='outer',on=df.index) # ran this code, dummify e
```

```
<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
```

[190]:
```
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
```

[191]:
```
df.head()
```

	key_0	Tweet	Platform	Emotion	I can't tell	Negative emotion	No e br
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

[192]:
```
df = df.rename(columns = {"I can't tell": "Uncertain", 'Negative emotion': 'Negat
                          , 'No emotion toward brand or product': 'No Emotion'
                          , 'Positive emotion':'Positive'})
```

[193]:
```
df = df.drop(columns='key_0')
df.head()
df.to_csv('Full_DF')
```

[76]:
```
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 upgrade.
 "@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its design.
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) &a
s)",
 '@teachntech00 New iPad Apps For #SpeechTherapy And Communication Are Showcased At The #SXSW Conferenc
M) #iear #edchat #asd',
 nan,
 '#SXSW is just starting, #CTIA is around the corner and #googleio is only a hop skip and a jump from t
roid fan',
 'Beautifully smart and simple idea RT @madebymany @thenextweb wrote about our #hollergram iPad app for
bit.ly/ieaVOB',)
 'Counting down the days to #sxsw plus strong Canadian dollar means stock up on Apple gear']
```

Tokenize the Words

[77]:
```
tokenz = word_tokenize(','.join(str(v) for v in corpus))
```

[78]:
```
tokenz[:10]
```

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

Create Stopwords List

[79]:
```
stopword_list = list(nlp.Defaults.stop_words)
len(nlp.Defaults.stop_words)
```

```
326
```

[80]:
```
stopword_list
```

```
 'my',
 'those',
 'were',
 'used',
 'myself',
 "'ll",
 'third',
 'beyond',
 "'re",
 '‘ll',
 'next',
 'go',
 'through',
 'except',
 'other',
 'and',
 'amongst',
 'few',
 'are',
 'one',
 'show',
 'thereupon',
 'would',
 'though',
```

[81]:
```
stopword_list.extend(string.punctuation)
```

[82]:
```
len(stopword_list)
```

```
 358
```

[83]:
```
stopword_list.extend(stopwords.words('english'))
```

[84]:
```
len(stopword_list)
```

```
 537
```

[85]:
```
additional_punc = ['“','”','...',"'",'’','``','https','rt','\.+']
stopword_list.extend(additional_punc)
stopword_list[-10:]
```

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

Remove stopwords and additional punctuation from the data

[86]:
```
stopped_tokenz = [word.lower() for word in tokenz if word.lower() not in stopworc
```

[87]:
```
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's misc

[88]:
```
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
stopword_list.extend(additional_misc)
stopword_list.extend(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'])
```

[89]:
```
lemmatizer = WordNetLemmatizer()
```

[90]:
```
clean_stopped_tokenz = [word.lower() for word in stopped_tokenz if word not in st
clean_lemmatized_tokenz = [lemmatizer.lemmatize(word.lower()) for word in stoppec
```

[91]:
```
freq_clean_lemma = FreqDist(clean_lemmatized_tokenz)
freq_lemma = freq_clean_lemma.most_common(5000)
freq_lemma2 = freq_clean_lemma.most_common(25)
```

[92]:
```
total_word_count = len(clean_lemmatized_tokenz)
```

[93]:
```
lemma_word_count = sum(freq_clean_lemma.values()) # just a number
```

[94]:
```
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 %
```

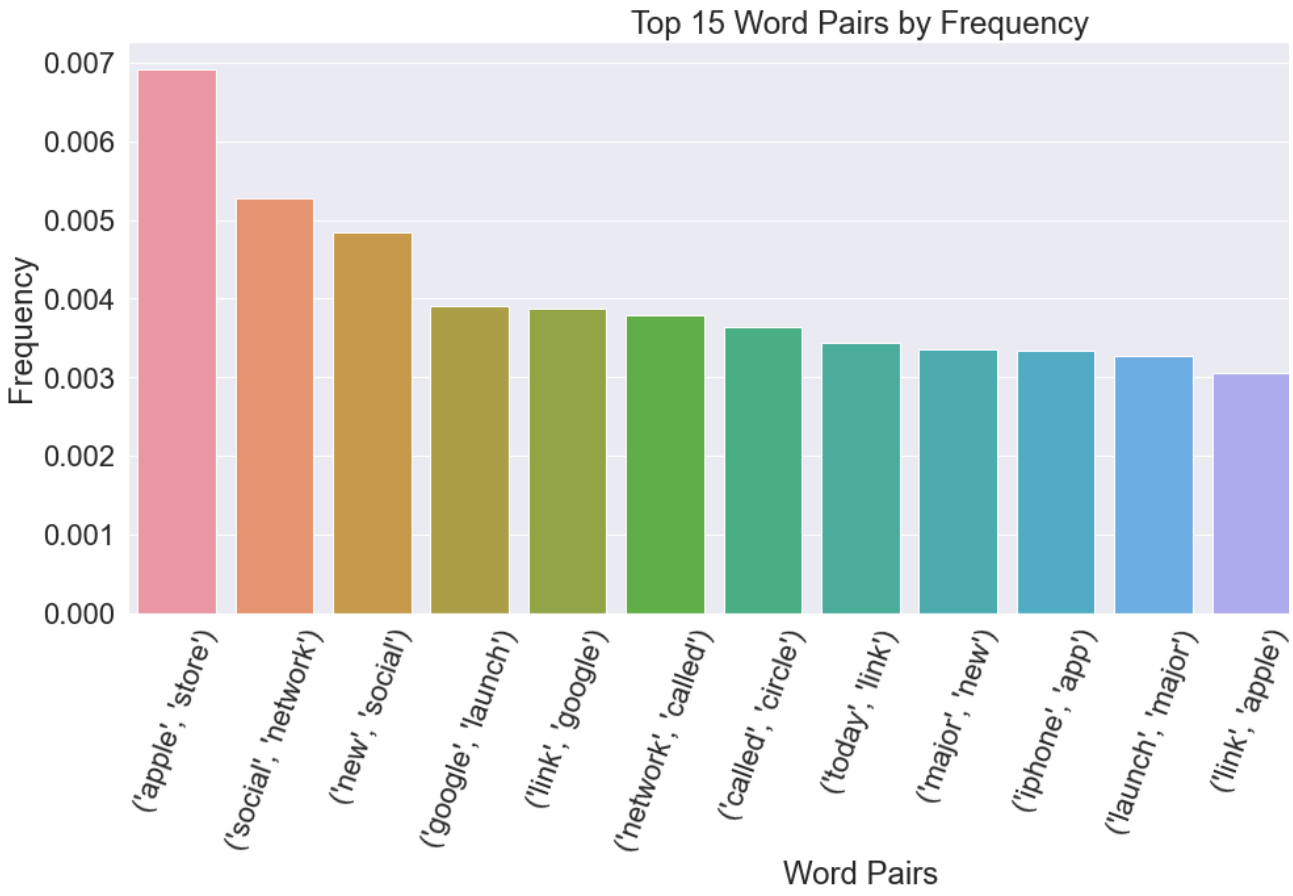
[95]:
```
# from wordcloud import WordCloud

# ## 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')
```

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

[97]:
```
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

[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=word_pairs.head(15)['Word'], y=word_pairs.head(15)['Freq'], pa
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()
```



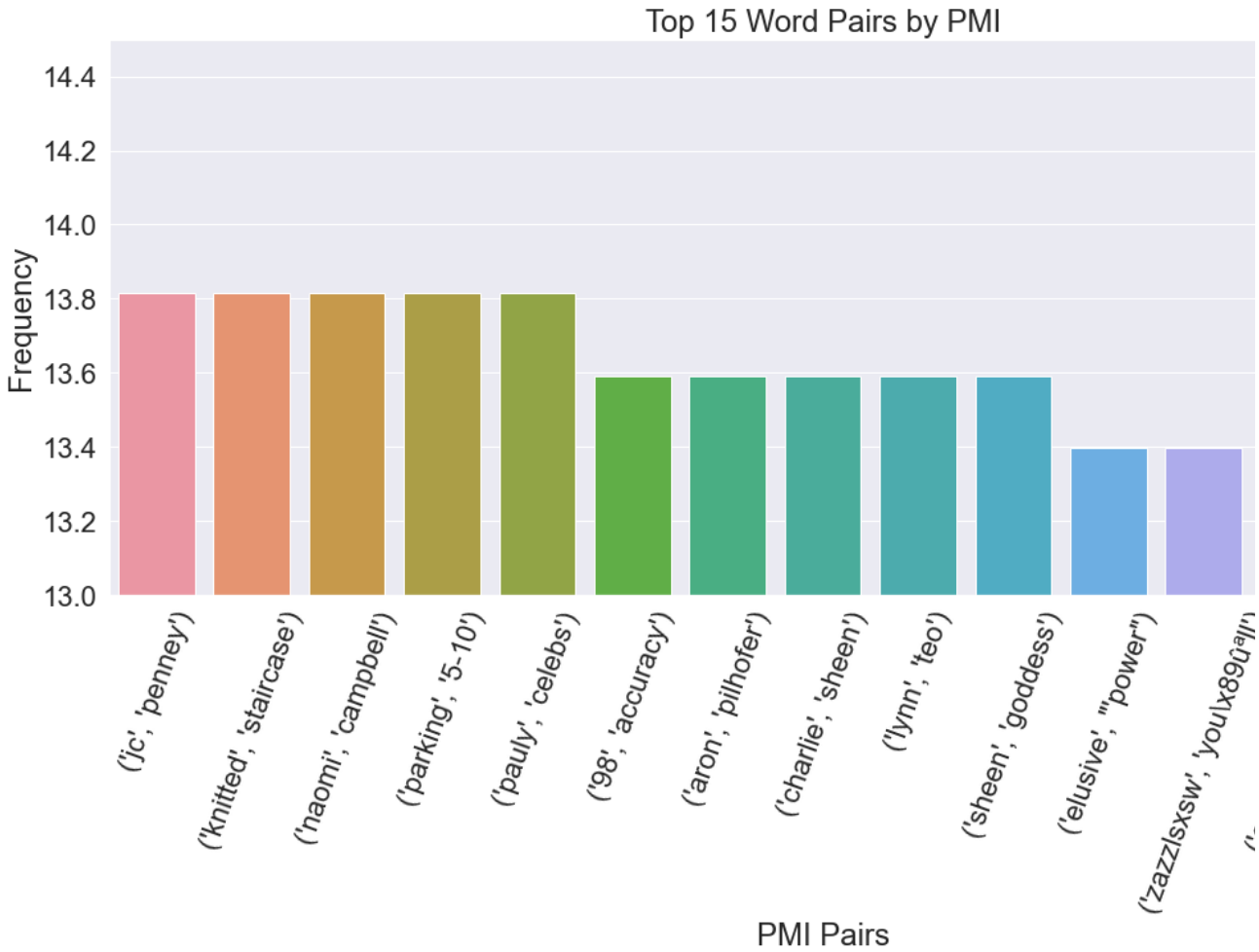
[99]:
```
tweet_pmi_finder = nltk.BigramCollocationFinder.from_words(clean_lemmatized_toker
tweet_pmi_finder.apply_freq_filter(5)

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

[272]:
```
PMI_list = pd.DataFrame(tweet_pmi_scored, columns=["Words","PMI"]).head(20)
PMI_list = PMI_list[PMI_list.PMI < 14]
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

[273]:

```
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'], palett
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()
```



[102]:
```
df1 = df
df.head()
```

	Tweet	Platform	Emotion	Uncertain	Nega
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

[103]:
```
df1 = df1.drop(columns=['Uncertain','No Emotion'])
# Turn negative and positive columns into one column of just negatives and positi
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_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

[104]:
```
df1.to_csv('Tweet.csv')
```

Create upsampled data, train and test sets

[105]:
```
from sklearn.utils import resample
```

[106]:
```
df_majority = df1.loc[df1['Positive_Bin']==1]
df_minority = df1.loc[df1['Positive_Bin']==0]
```

[107]:
```
df_minority.shape
```

```
(570, 4)
```

[108]:
```
df_majority.shape
```

```
(2978, 4)
```

[109]:
```
df_min_sample = resample(df_minority, replace=True, n_samples=1000, random_state=
```

[110]:
```
df_maj_sample = resample(df_majority, replace=True, n_samples=2500, random_state=
```

[111]:
```
df_upsampled = pd.concat([df_min_sample, df_maj_sample], axis=0)
df_upsampled.shape
```

```
(3500, 4)
```

[112]:
```
X, y = df_upsampled['Tweet'], df_upsampled['Positive_Bin']
```

[113]:
```
df_upsampled.to_csv('Upsampled.csv')
```

Train/Test Split

[114]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

[115]:
```
scaler_object = MaxAbsScaler()
```

[116]:
```
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
```

[117]:
```
y_train.value_counts(0)
y_test.value_counts(1)
```

```
2020-12-17 14:41:18,922 : 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

[118]:
```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer, Tfi
from sklearn.ensemble import RandomForestClassifier

tokenizer = nltk.TweetTokenizer(preserve_case=False)

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

[119]:
```
# for row in X_train:
#     for word in row:
#         lemmatizer.lemmatize(X_train[row][word])
# return X_train[word][row]
```

[120]:
```
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 st
with your preprocessing. Tokenizing the stop words generated tokens [":'[", ':/', 'a-z', 'a-za-z', 'htt
p_words.
  warnings.warn('Your stop_words may be inconsistent with '
```

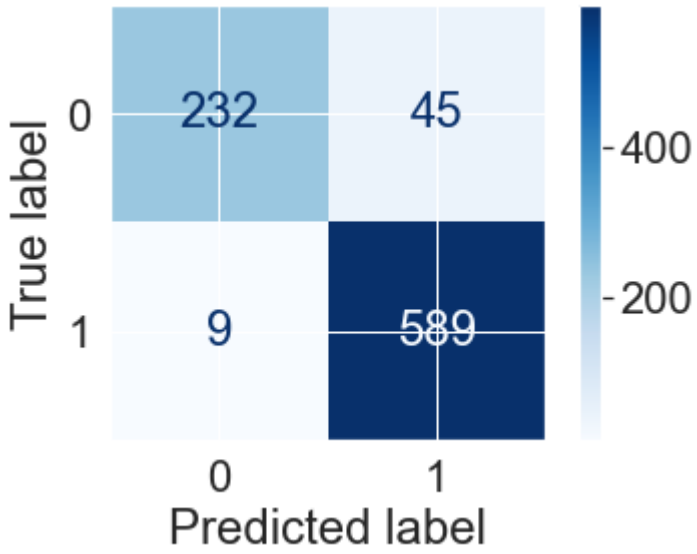
[121]:
```
ran_for = RandomForestClassifier(class_weight='balanced')
model = ran_for.fit(X_train_count, y_train)
```

[122]:
```
y_hat_test = model.predict(X_test_count)
```

Evaluate Models

[123]:
```
evaluate_model(y_test, y_hat_test, X_test_count,clf=model) # 1 denotes Positive T
```

	precision	recall	f1-score	support
0	0.96	0.84	0.90	277
1	0.93	0.98	0.96	598
accuracy			0.94	875
macro avg	0.95	0.91	0.93	875
weighted avg	0.94	0.94	0.94	875



[124]:
```
tf_idf_vectorizer = TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                    stop_words=stopword_list,decode_error='ignore
```

[125]:
```
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 st
with your preprocessing. Tokenizing the stop words generated tokens [":'[", ':/', 'a-z', 'a-za-z', 'htt
p_words.
  warnings.warn('Your stop_words may be inconsistent with '


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

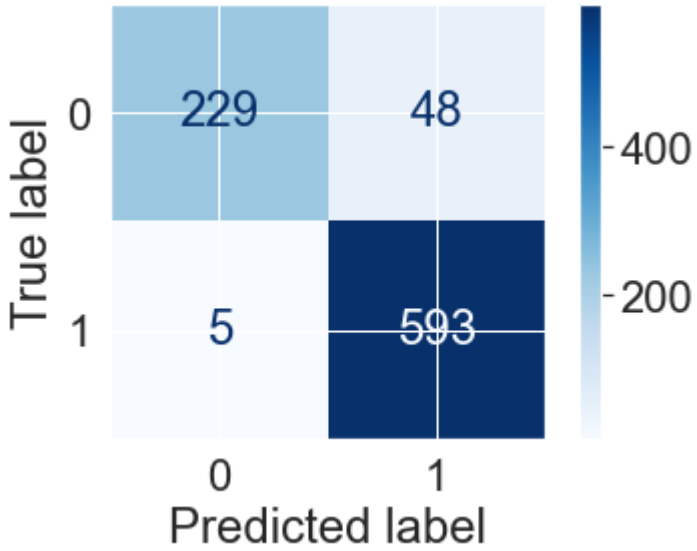
[126]:
```
from sklearn.ensemble import RandomForestClassifier
```

[127]:
```
ran_for = RandomForestClassifier(class_weight='balanced')
model_tf_idf = ran_for.fit(X_train_tf_idf,y_train)
```

[128]:
```
y_hat_tf_idf = model_tf_idf.predict(X_test_count)
```

[129]:
```
evaluate_model(y_test, y_hat_tf_idf, X_test_tf_idf,clf=model_tf_idf) # slightly b
```

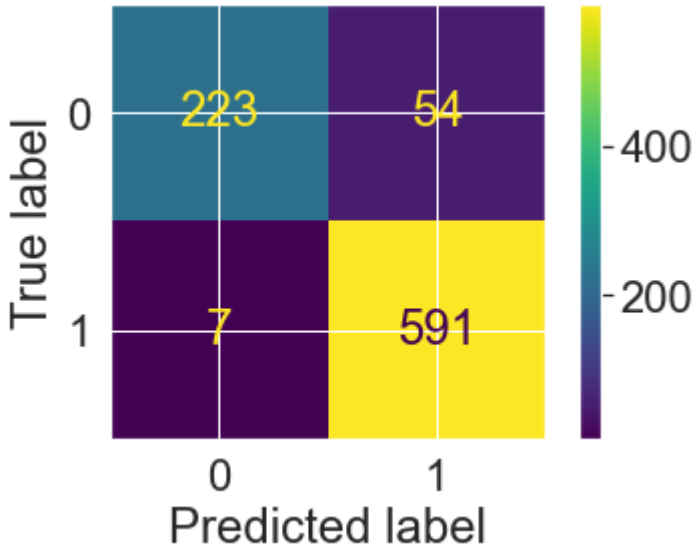
```
              precision    recall  f1-score   support

           0       0.95      0.57      0.71       277
           1       0.83      0.98      0.90       598

    accuracy                           0.85       875
   macro avg       0.89      0.78      0.81       875
weighted avg       0.87      0.85      0.84       875
```



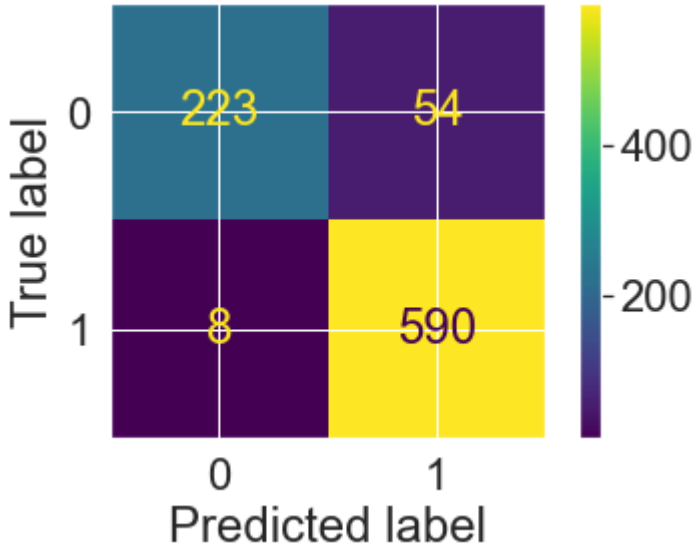
[130]:

```
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)
```

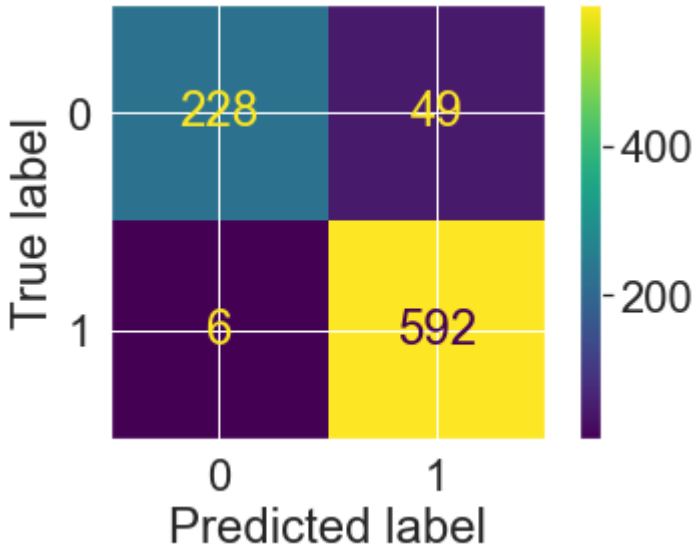
```
Accuracy Score:  0.9302857142857143
Precision Score:  0.9162790697674419
Recall Score:  0.9882943143812709
F1 Score:  0.9509251810136766
RandomForestClassifier()   0.9302857142857143
```




```
Accuracy Score:  0.9291428571428572
Precision Score:  0.9161490683229814
Recall Score:  0.9866220735785953
F1 Score:  0.9500805152979066
RandomForestClassifier()   0.9291428571428572
```

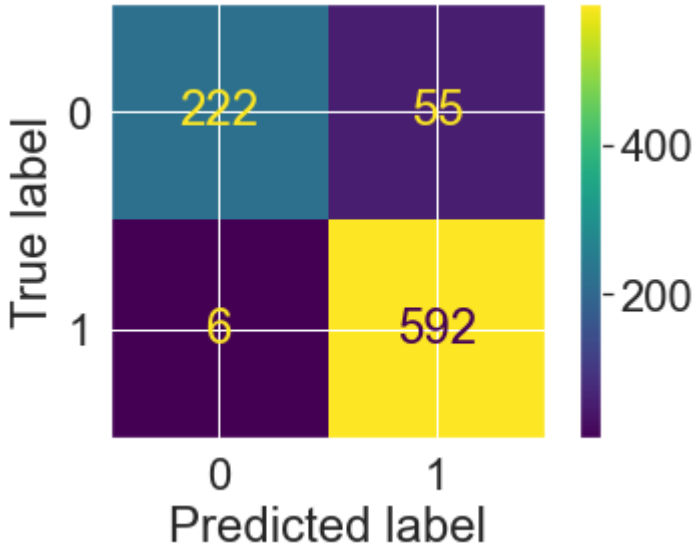



```
Accuracy Score:  0.9371428571428572
Precision Score:  0.9235569422776911
Recall Score:  0.9899665551839465
F1 Score:  0.9556093623890234
RandomForestClassifier()   0.9371428571428572
```

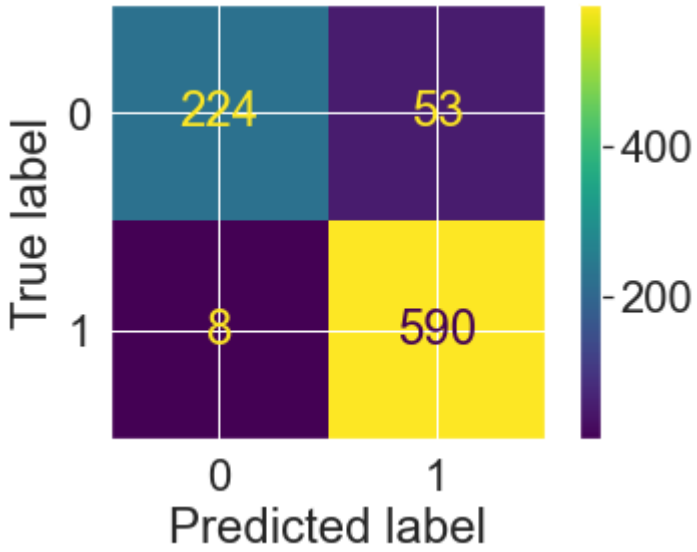


[131]:
```
for model in models:
    single_model_opt(ran_for, X_train_tf_idf, y_train, X_test_tf_idf, y_test)
```

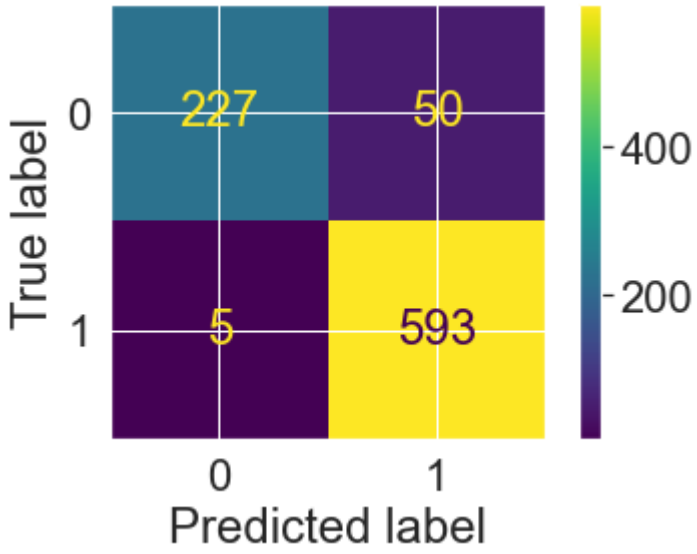
```
Accuracy Score:  0.9302857142857143
Precision Score:  0.9149922720247295
Recall Score:  0.9899665551839465
F1 Score:  0.9510040160642571
RandomForestClassifier()   0.9302857142857143
```




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



```
Accuracy Score:  0.9371428571428572
Precision Score:  0.9222395023328149
Recall Score:  0.9916387959866221
F1 Score:  0.9556809024979854
RandomForestClassifier()   0.9371428571428572
```

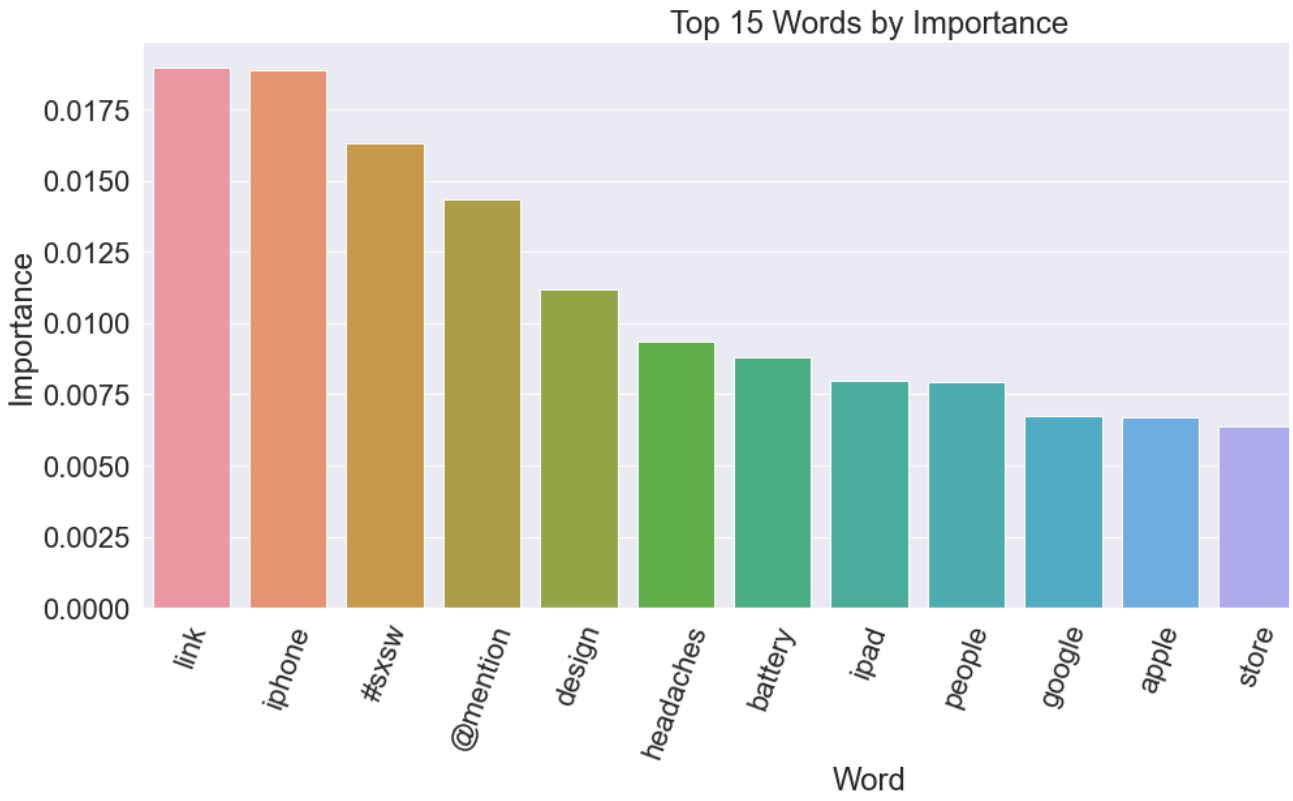


[132]:
```
tf_idf_vectorizer.get_feature_names()
```

```
['##sxsw',
 '#10',
 '#106',
 '#11ntc',
 '#1406-08',
 '#15slides',
 '#310409h2011',
 '#4sq',
 '#911tweets',
 '#abacus',
 '#accesssxsw',
 '#accordion',
 '#aclu',
 '#adam',
 '#addictedtotheinterwebs',
 '#adpeopleproblems',
 '#agchat',
 '#agileagency',
 '#agnerd',
 '#allhat3',
 '#alwayshavingtoplugin',
 '#amateurhour',
 '#android',
 "#android's",
```

[133]:
```
importance = pd.Series(ran_for.feature_importances_,index=tf_idf_vectorizer.get_f
importance = pd.DataFrame(importance).sort_values(by=0,ascending=False)
```

[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=importance.head(15).index, y=importance.head(15)[0], palette=p
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()
```



[135]:
```
vectorizer = CountVectorizer()
tf_transform = TfidfTransformer(use_idf=True)
```

[136]:
```
text_pipe = Pipeline(steps=[
    ('count_vectorizer',vectorizer),
    ('tf_transformer',tf_transform)])
```

[137]:
```
RandomForestClassifier(class_weight='balanced')
```

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

[138]:
```
full_pipe = Pipeline(steps=[
    ('text_pipe',text_pipe),
    ('clf',RandomForestClassifier(class_weight='balanced'))
])
```

[139]:
```
X_train_pipe = text_pipe.fit_transform(X_train)
```

[140]:
```
X_test_pipe = text_pipe.transform(X_test)
```

[141]:
```
X_train_pipe
```

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

[142]:
```
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']}
```

[143]:

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

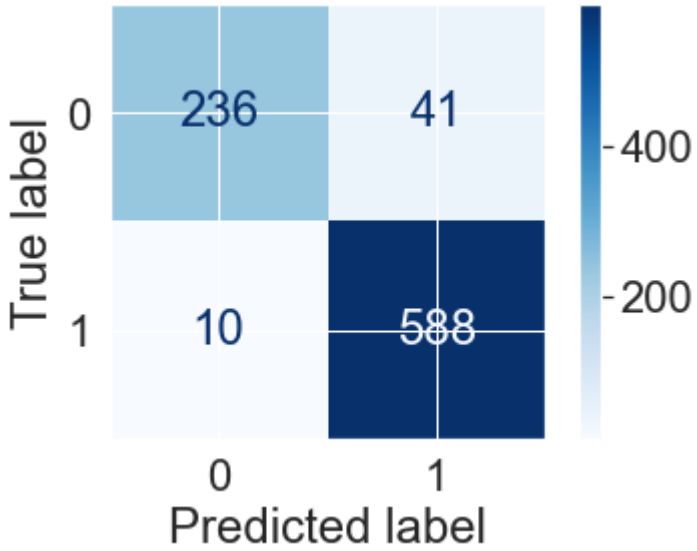
```
C:\Users\josep\anaconda3\lib\site-packages\sklearn\feature_extraction\text.py:383: UserWarning: Your st
with your preprocessing. Tokenizing the stop words generated tokens [":'[", ':/', 'a-z', 'a-za-z', 'htt
op_words.
  warnings.warn('Your stop_words may be inconsistent with '
C:\Users\josep\anaconda3\lib\site-packages\sklearn\feature_extraction\text.py:383: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

[144]:

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

[145]:
```
evaluate_model(y_test,y_hat_test,X_test,best_pipe)
```

```
              precision    recall  f1-score   support

           0       0.96      0.85      0.90       277
           1       0.93      0.98      0.96       598

    accuracy                           0.94       875
   macro avg       0.95      0.92      0.93       875
weighted avg       0.94      0.94      0.94       875
```



[146]:
```
X_train_pipe.shape
```

```
(2625, 4256)
```

[147]:
```
features = text_pipe.named_steps['count_vectorizer'].get_feature_names()
features[:10]
```

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

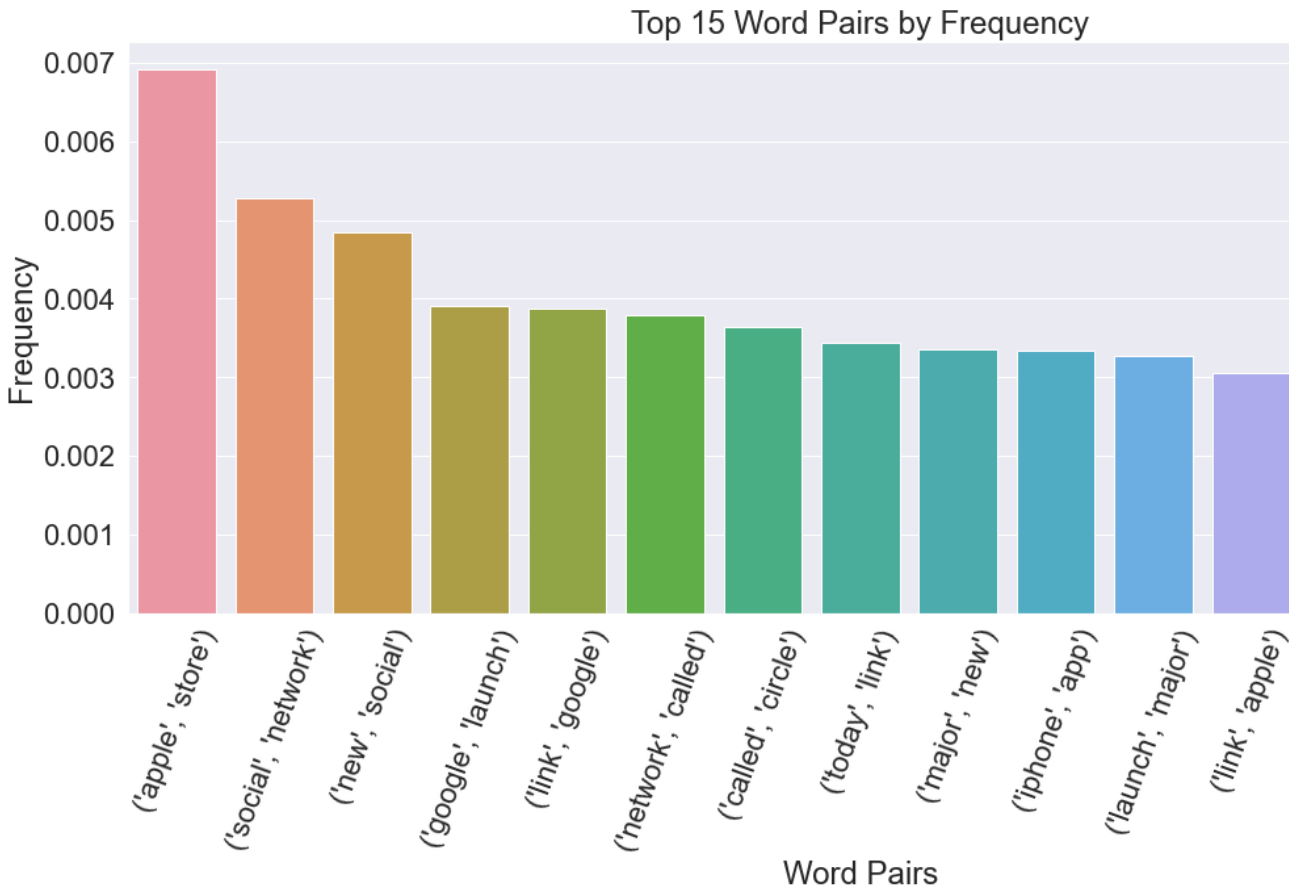
[148]:
```
bigram_measures = nltk.collocations.BigramAssocMeasures()
tweet_finder = nltk.BigramCollocationFinder.from_words(clean_lemmatized_tokenz)
tweets_scored = tweet_finder.score_ngrams(bigram_measures.raw_freq)
```

[149]:
```
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

[150]:
```
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
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 (binary)

[151]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import models, layers, optimizers
```

[152]:
```
model = 0
```

[198]:
```
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'>
```

[257]:
```
one_hot_results = tokenizer.texts_to_matrix(tweets, mode='binary')
print('one_hot_results type:', type(one_hot_results))
one_hot_results
```

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

```
array([[0., 1., 0., ..., 0., 0., 0.],
       [0., 1., 1., ..., 0., 0., 0.],
       [0., 1., 1., ..., 0., 0., 0.],
       ...,
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.]])
```

[200]:
```
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

```
Found 4816 unique tokens.
```

[201]:
```
print('Dimensions of our coded results:', np.shape(one_hot_results))
```

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

[236]:
```
print(y.shape)
print(one_hot_results.shape)
```

```
(3500, 1)
(3500, 10000)
```

[204]:
```
import random
```

[239]:
```
random.seed(42)
test_index = random.sample(range(1,3500), 1500)

test = one_hot_results[test_index]
train = np.delete(one_hot_results, test_index, 0) #.astype('float32').reshape((-1

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: (1500, 1)
Train label shape: (2000, 1)
Test shape: (1500, 10000)
Train shape: (2000, 10000)
```

[238]:
```
# tokenizer = Tokenizer()
# tokenizer.fit_on_texts(list(X))
# sequences = tokenizer.texts_to_sequences(X)
# X = sequence.pad_sequences(sequences,maxlen=100)
```

[240]:
```
tokenizer.word_counts
```

```
('in', 711),
('app', 464),
('store', 536),
('includes', 23),
('uberguide', 23),
('to', 1439),
('link', 1152),
('2011', 27),
('novelty', 19),
('news', 67),
('apps', 103),
('fades', 21),
('fast', 42),
('among', 23),
('digital', 37),
('delegates', 19),
('rule', 2),
('no', 161),
('more', 102),

('ooing', 2),
('and', 636),
('ahing', 2),
('over', 68),
('your', 168),
```

[207]:
```
vocab_size = len(tokenizer.word_counts)
seq_len = X.shape[1]
```

[212]:
```
print(vocab_size)
print(seq_len)
```

```
4816
100
```

[209]:
```
print(type(X),X.shape)
print(type(y),y.shape)
```

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

[210]:
```
X = np.asarray(X).astype('float32')
```

[241]:
```
print(type(X),X.shape)
print(type(y),y.shape)
```

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

[252]:
```
# Initialize a sequential 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(2, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['acc'])
```

[253]:
```
model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_27 (Dense)             (None, 32)                320032
_________________________________________________________________
dense_28 (Dense)             (None, 16)                528
_________________________________________________________________
dense_29 (Dense)             (None, 2)                 34
=================================================================
Total params: 320,594
Trainable params: 320,594
Non-trainable params: 0
_________________________________________________________________
```

[254]:
```
train.shape
```

```
(2000, 10000)
```

[255]:
```
label_train.shape
```

```
(2000, 1)
```

[256]:
```
history = model.fit(train,label_train, batch_size=32, epochs=10, verbose=2, valic
```

```
Epoch 1/10
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-256-c75e5aa679b8> in <module>
----> 1 history = model.fit(train,label_train, batch_size=32, epochs=10, verbose=2, validation_split=.2

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\training.py in fit(self,
rbose, callbacks, validation_split, validation_data, shuffle, class_weight, sample_weight, initial_epoc
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__(sel
    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(self,
    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 _initialize(
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: disable
    726             *args, **kwds))
    727

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\function.py in _get_concrete_fu
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_fu
   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_fu
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_graph
nc, args, kwargs, signature, func_graph, autograph, autograph_options, add_control_dependencies, arg_na
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_fn(*
    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(*a
    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\trainin
        return step_function(self, iterator)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\trainin
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\distribute\distribut
        return self._extended.call_for_each_replica(fn, args=args, kwargs=kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\distribute\distribut
eplica
        return self._call_for_each_replica(fn, args, kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\distribute\distribut
replica
        return fn(*args, **kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\trainin
        outputs = model.train_step(data)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\trainin
        loss = self.compiled_loss(
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\compile
        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:152
        losses = call_fn(y_true, y_pred)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\losses.py:256
        return ag_fn(y_true, y_pred, **self._fn_kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\dispatch.py:201
        return target(*args, **kwargs)
```

```
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\losses.py:1608
        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:201
        return target(*args, **kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\backend.py:497
        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:201
        return target(*args, **kwargs)
    C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\ops\nn_impl.py:173 s
its
        raise ValueError("logits and labels must have the same shape (%s vs %s)" %

    ValueError: logits and labels must have the same shape ((32, 2) vs (32, 1))
```

n [ ]:

Deep NLP using Word2Vec

[166]:
```
from nltk import word_tokenize
```

[167]:
```
data = df_upsampled['Tweet'].map(word_tokenize)
```

[168]:
```
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
```

[169]:
```
model_W2V = Word2Vec(data, size =100, window=5, min_count=1, workers=4)
```

```
2020-12-17 14:49:38,217 : INFO : collecting all words and their counts
2020-12-17 14:49:38,218 : INFO : PROGRESS: at sentence #0, processed 0 words, keeping 0 word types
2020-12-17 14:49:38,235 : INFO : collected 5920 word types from a corpus of 86715 raw words and 3500 se
2020-12-17 14:49:38,236 : INFO : Loading a fresh vocabulary
2020-12-17 14:49:38,246 : INFO : effective_min_count=1 retains 5920 unique words (100% of original 5920
2020-12-17 14:49:38,247 : INFO : effective_min_count=1 leaves 86715 word corpus (100% of original 86715
2020-12-17 14:49:38,263 : INFO : deleting the raw counts dictionary of 5920 items
2020-12-17 14:49:38,264 : INFO : sample=0.001 downsamples 52 most-common words
2020-12-17 14:49:38,265 : INFO : downsampling leaves estimated 56808 word corpus (65.5% of prior 86715)
2020-12-17 14:49:38,278 : INFO : estimated required memory for 5920 words and 100 dimensions: 7696000 b
2020-12-17 14:49:38,279 : INFO : resetting layer weights
2020-12-17 14:49:39,345 : INFO : training model with 4 workers on 5920 vocabulary and 100 features, usi
ative=5 window=5
2020-12-17 14:49:39,406 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,408 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,410 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,413 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,414 : INFO : EPOCH - 1 : training on 86715 raw words (56803 effective words) took 0
s
2020-12-17 14:49:39,451 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,455 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,459 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,460 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,460 : INFO : EPOCH - 2 : training on 86715 raw words (56660 effective words) took 0
s
2020-12-17 14:49:39,497 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,500 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,503 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,505 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,506 : INFO : EPOCH - 3 : training on 86715 raw words (56731 effective words) took 0
s
2020-12-17 14:49:39,545 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,549 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,550 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,552 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,553 : INFO : EPOCH - 4 : training on 86715 raw words (56764 effective words) took 0
s
2020-12-17 14:49:39,588 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,594 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,596 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,598 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,598 : INFO : EPOCH - 5 : training on 86715 raw words (56899 effective words) took 0
s
2020-12-17 14:49:39,599 : INFO : training on a 433575 raw words (283857 effective words) took 0.3s, 112
```

[170]:
```
model_W2V.train(data,total_examples=model_W2V.corpus_count, epochs=10)
```

```
2020-12-17 14:49:39,603 : WARNING : Effective 'alpha' higher than previous training cycles
2020-12-17 14:49:39,604 : INFO : training model with 4 workers on 5920 vocabulary and 100 features, usi
ative=5 window=5
2020-12-17 14:49:39,650 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,655 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,660 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,662 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,663 : INFO : EPOCH - 1 : training on 86715 raw words (56819 effective words) took 0
s
2020-12-17 14:49:39,705 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,712 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,714 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,715 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,716 : INFO : EPOCH - 2 : training on 86715 raw words (56894 effective words) took 0
s
2020-12-17 14:49:39,752 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,755 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,759 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,760 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,761 : INFO : EPOCH - 3 : training on 86715 raw words (56831 effective words) took 0
s
2020-12-17 14:49:39,800 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,806 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,807 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,808 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,809 : INFO : EPOCH - 4 : training on 86715 raw words (56898 effective words) took 0
s
2020-12-17 14:49:39,845 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,849 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,852 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,854 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,855 : INFO : EPOCH - 5 : training on 86715 raw words (56933 effective words) took 0
s
2020-12-17 14:49:39,893 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,895 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,899 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,900 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,901 : INFO : EPOCH - 6 : training on 86715 raw words (56743 effective words) took 0
s
2020-12-17 14:49:39,940 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,946 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,949 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:39,950 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:39,951 : INFO : EPOCH - 7 : training on 86715 raw words (56833 effective words) took 0
s
2020-12-17 14:49:39,990 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:39,996 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:39,997 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:40,000 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:40,000 : INFO : EPOCH - 8 : training on 86715 raw words (56803 effective words) took 0
s
2020-12-17 14:49:40,039 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:40,041 : INFO : worker thread finished; awaiting finish of 2 more threads
```

```
2020-12-17 14:49:40,044 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:40,045 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:40,046 : INFO : EPOCH - 9 : training on 86715 raw words (56790 effective words) took 0
s
2020-12-17 14:49:40,086 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 14:49:40,088 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 14:49:40,093 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 14:49:40,093 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 14:49:40,094 : INFO : EPOCH - 10 : training on 86715 raw words (56776 effective words) took
s/s
2020-12-17 14:49:40,094 : INFO : training on a 867150 raw words (568320 effective words) took 0.5s, 116


(568320, 867150)
```

[171]:
```
wv = model_W2V.wv
```

[267]:
```
wv.most_similar(positive='phone')
```

```
[('moment-it', 0.9723027348518372),
 ('nor', 0.9712374806404114),
 ('tweeted', 0.9692162275314331),
 ('3g', 0.9681060314178467),
 ('-Google', 0.9630390405654907),
 ('horrendous', 0.9612216353416443),
 ('dawdled', 0.9598613381385803),
 ('correcting', 0.9595307111740112),
 ('My', 0.9591711759567261),
 ('Qrank', 0.9568673372268677)]
```

[173]:
```
wv['help']
```

```
array([-0.31967285, -0.17885478,  0.01392739, -0.20652783, -0.24002904,
        0.05603355,  0.23169126, -0.11376803, -0.11942402, -0.29525623,
        0.3290574 ,  0.06339365,  0.3104117 ,  0.05134623,  0.12008827,
       -0.2247823 ,  0.01781183, -0.1464717 , -0.1455513 ,  0.07288111,
       -0.03163346,  0.29379946, -0.00203749,  0.02973694, -0.2917498 ,
       -0.28070888, -0.26782623,  0.10527655, -0.14054094, -0.03771594,
        0.33188355, -0.02599237, -0.0525349 ,  0.05000544,  0.04384491,
        0.19176967, -0.04553479, -0.08937339, -0.02473517,  0.01382217,
       -0.0907728 ,  0.28192258,  0.19038127,  0.00607586, -0.01968819,
        0.0785262 ,  0.21970062,  0.28426826,  0.10126912,  0.14359671,
        0.05886083, -0.18110804,  0.18728036, -0.19307703, -0.0777308 ,
        0.25104517, -0.47962093,  0.13631037,  0.00184456,  0.01349466,
       -0.1595733 ,  0.25049472,  0.12245066,  0.2686916 ,  0.02174757,
        0.31893703,  0.11131237,  0.01023629,  0.01475756, -0.0240675 ,
       -0.19176066, -0.18991126,  0.24131042, -0.33164704,  0.17345098,
       -0.01427521, -0.20412044, -0.10288385,  0.05892187, -0.12293504,
       -0.03255542, -0.09149769,  0.1287596 , -0.13189872, -0.07963987,
       -0.23899263,  0.10492894, -0.09980745, -0.04041791, -0.14108348,
        0.05543073, -0.10543934, -0.08044261,  0.47764343,  0.19938034,
       -0.1042861 ,  0.3239305 , -0.32515568, -0.02896872,  0.25202996],
      dtype=float32)
```

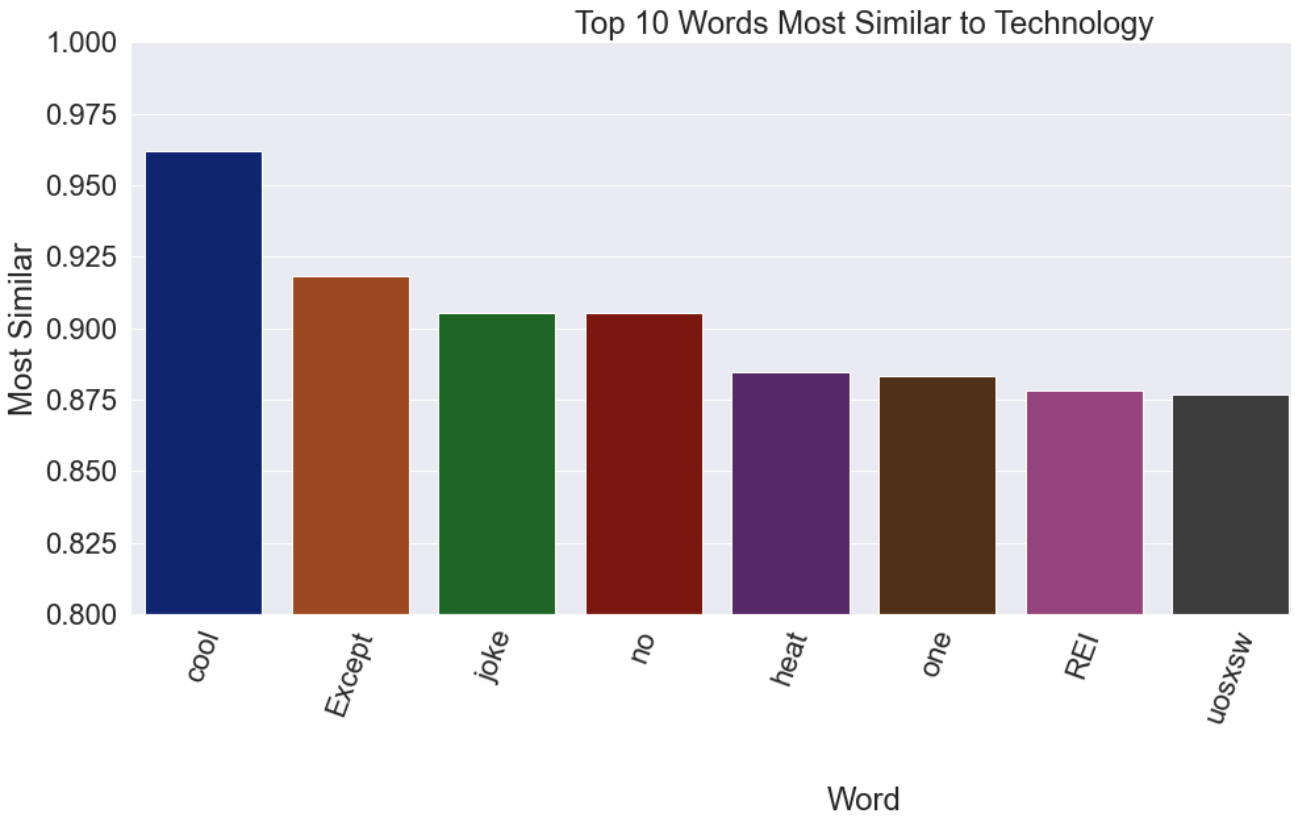
[174]:
```
wv.vectors
```

```
array([[-0.22285825, -0.9334564 , -0.49763873, ..., -0.39693695,
        -0.4778785 ,  0.5305556 ],
       [-0.646233  , -0.7081872 , -0.23683992, ..., -0.5656346 ,
        -0.08979444,  0.38222355],
       [-0.01102781, -0.7331643 , -0.31039104, ...,  0.25689587,
        -1.2840519 ,  0.3224538 ],
       ...,
       [-0.03657845, -0.04447062, -0.0088504 , ..., -0.0449322 ,
        -0.02913979,  0.0439421 ],
       [-0.00787574,  0.02462851, -0.01310325, ...,  0.00459611,
        -0.03039238,  0.0015835 ],
       [-0.02813163,  0.00890381, -0.00215271, ..., -0.01855575,
        -0.00444138,  0.0224769 ]], dtype=float32)
```

[280]:
```
df_tech = pd.DataFrame(wv.most_similar(positive=['technology']))
```

[282]:

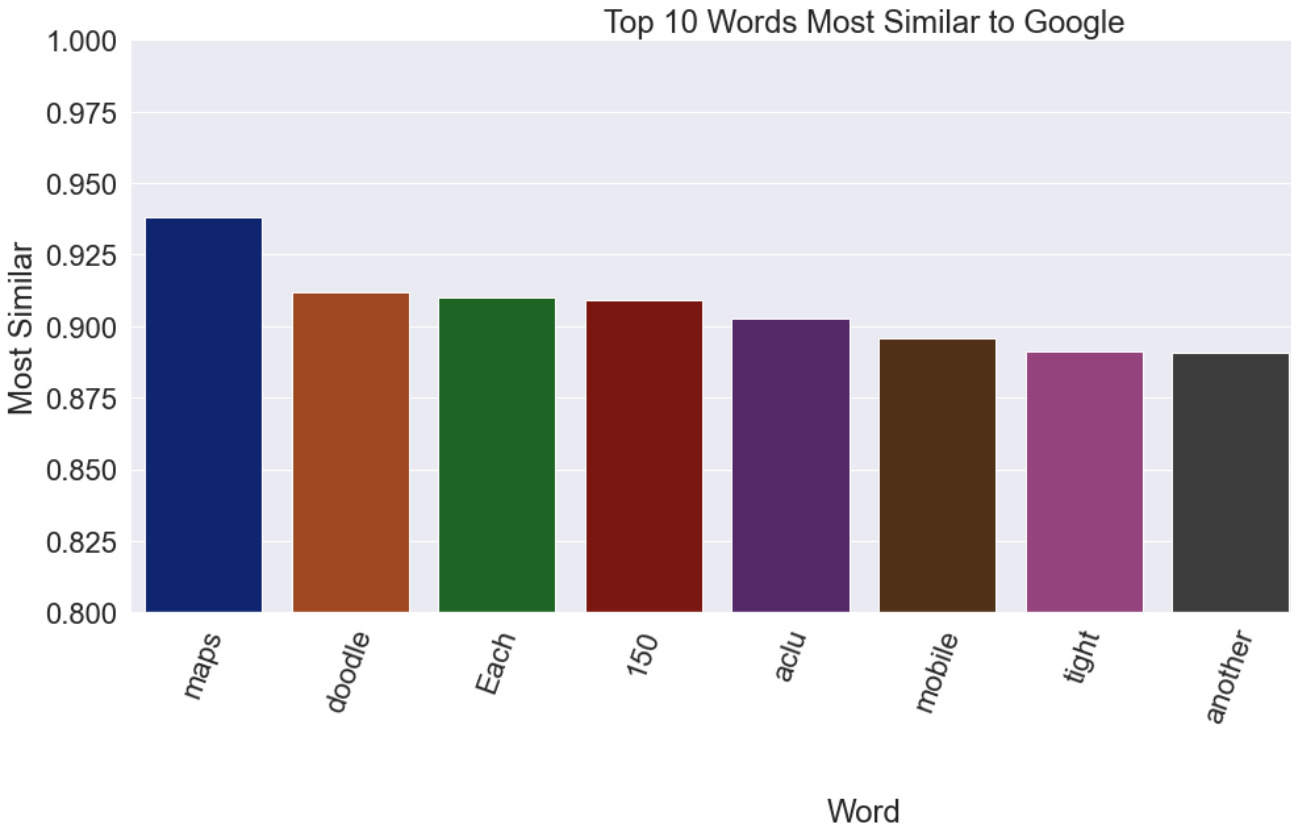
```
fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=df_tech.head(10)[0], y=df_tech.head(10)[1], palette=palette)
ax.set(xlabel="Word",ylabel="Most Similar")
plt.ticklabel_format(style='plain',axis='y')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



[283]:
```
df_google = pd.DataFrame(wv.most_similar(positive=['google']))
df_google
```

	0	1
0	maps	0.938149
1	doodle	0.911753
2	Each	0.910172
3	150	0.909193
4	aclu	0.902651
5	mobile	0.895684
6	tight	0.891417
7	another	0.890622
8	unpaid	0.888086
9	Wowwwwww	0.887056

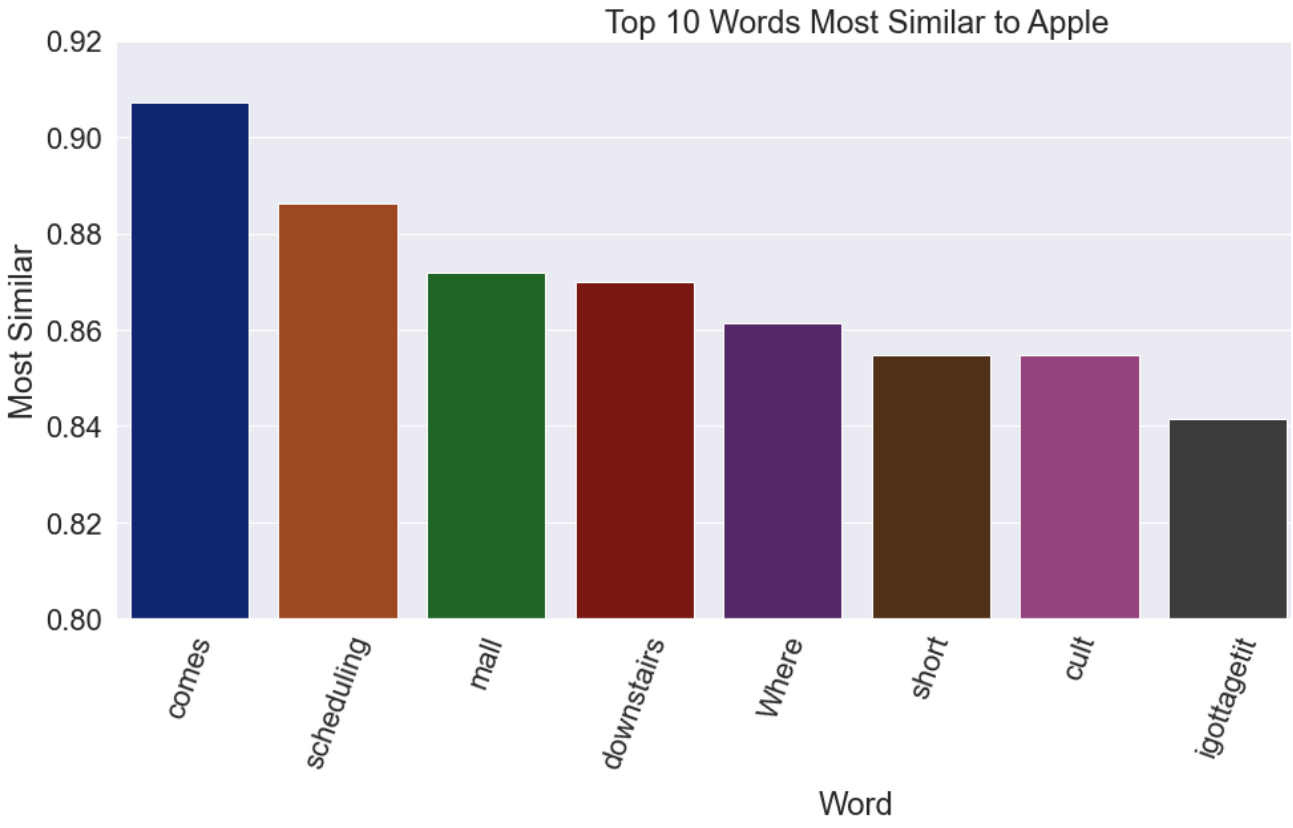
[284]:
```
fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=df_google.head(10)[0], y=df_google.head(10)[1], palette=palett
ax.set(xlabel="Word",ylabel="Most Similar")
plt.ticklabel_format(style='plain',axis='y')
plt.ylim(.8,1)
plt.xticks(rotation=70)
plt.title('Top 10 Words Most Similar to Google')
plt.show()
```



[285]:
```
df_apple = pd.DataFrame(wv.most_similar(positive=['apple']))
df_apple
```

	0	1
0	comes	0.907297
1	scheduling	0.886354
2	mall	0.871902
3	downstairs	0.870072
4	Where	0.861456
5	short	0.854704
6	cult	0.854672
7	igottagetit	0.841427
8	nerdheaven	0.838491
9	Ave	0.836979

[289]:
```
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=palette)
ax.set(xlabel="Word",ylabel="Most Similar")
plt.ticklabel_format(style='plain',axis='y')
plt.ylim(.8,.92)
plt.xticks(rotation=70)
plt.title('Top 10 Words Most Similar to Apple')
plt.show()
```



n [ ]: