

EDA, Preprocessing, and Tweet Analysis Notebook

```
In [2]: import numpy as np
import pandas as pd
import spacy
import re
import nltk
import matplotlib.pyplot as plt
import logging

logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)

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

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

from sklearn.metrics import classification_report, confusion_matrix, plot_confusion_matrix
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.naive_bayes import BernoulliNB, CategoricalNB, GaussianNB, MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer,
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 [3]: nlp = spacy.load('en_core_web_sm')
```

```
In [4]: print(stopwords)
print(nlp.Defaults.stop_words)
```

```
<WordListCorpusReader in '.../corpora/stopwords' (not loaded yet)>
{'seems', 'which', 'thereupon', 'really', 'except', 'ourselves', 'third', 'those', 'never', 'more',
'o', 've', 'whom', 'nothing', 'my', 'm', 'at', 'several', 'eleven', 'part', 'seem', 'when', 'when',
ever', 'quite', 'thus', 'could', 'll', 'again', 'just', 'these', 'see', 'via', 'being', 'over',
h', 'into', 'and', 'around', 'rather', 'n't', 'unless', 'moreover', 'else', 'sometimes', 'his', 'is',
n', 'where', 'amongst', 'beyond', 'through', 'empty', 'somewhere', 'became', 'other', 'he', 'same',
rds', 'though', 'five', 'that', 'was', 'call', 'each', 'yourselves', 'all', 'then', 'whence', 'six',
y', 'per', 'between', 'someone', 'still', 'six', 'neither', 'keep', 'indeed', 'than', 'twelve', '
'everywhere', 'whereas', 'down', 'further', 'whole', 'of', 'whenever', 'top', 'anyone', 'to', 'to',
'here', 'whose', 'hundred', 'latterly', 'fifteen', 'many', 'mostly', 'who', 'serious', 'ca', 'now',
n', 'bottom', 'even', 'every', 'might', 'former', 'various', 'since', 'either', 've', 'once', 'st',
y', 'you', 'whatever', 'hereupon', 'has', 'have', 'four', 'ours', 'say', 'will', 'seemed', 'anyhow',
n', 'why', 'even', 'amount', 'cannot', 'or', 'although', 'there', 'itself', 'give', 'can', 'make',
ays', 'are', 'may', 'name', 'd', 'what', 'namely', 'eight', 'after', 'along', 'anywhere', 'made',
ers', 'forty', 'by', 'some', 'three', 'becomes', 'becoming', 's', 'herein', 'during', 'but', 'wh',
ts', 'almost', 'anything', 's', 'from', 'using', 'done', 'an', 've', 'both', 'off', 'therein',
ur', 're", 'own', 'him', 'll", 'does', 'been', 'besides', 'whether', 'her', 'before', 'last', 't',
ore', 'their', 'move', 'nine', 'anyway', 'hence', 'alone', 'myself', 'how', 'n't", 'd', 'through',
y', 'me', 'thru', 'himself', 'must', 'up', 's", 'about', 'on', 'please', 'across', 'towards', 'ge',
y', 'would', 'should', 'too', 're', 'we', 'nobody', 'under', 'noone', 'yourself', 'with', 'often',
g', 'otherwise', 'am', 'well', 'back', 'thence', 'hereafter', 'take', 'not', 'put', 'least', 'ever',
"d", 'due', 'whereby', 'another', 'already', 'us', 'thereafter', 'until', 're', 'only', 'few',
'your', 'also', 'such', 'nevertheless', 'much', 'be', 'seeming', 'meanwhile', 'mine', 're', 'one',
irst'}
```

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

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

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_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	@sxsxw 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 [7]: 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 [8]: df = df.rename(columns= {'is_there_an_emotion_directed_at_a_brand_or_product': 'Emotion', 'emotion_in_tweet_is_directed_at': 'Platform'})
```

```
In [9]: df = df.rename(columns= {'tweet_text': 'Tweet'})
```

```
In [10]: 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	@sxsxw 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 [11]: df_dummify = pd.get_dummies(df['Emotion'])
```

```
In [12]: 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 [13]: 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 [14]: df.info()

df = pd.merge(df, df_dummify, how='outer', on=df.index) # ran this code, dumm
```

```

<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 [15]:

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 [16]:

df.head()

key_0		Tweet	Platform	Emotion	I can't tell	Negative emotion	
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 [17]:

```

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

```

```
In [18]: df = df.drop(columns='key_0')
df.head()
```

	Tweet	Platform	Emotion	Uncertain
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	0
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive emotion	0
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	0

```
In [19]: 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 upl
"@jessedee Know about @fludapp ? Awesome iPad/iPhone app that you'll likely appreciate for its d
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/conferenc
s)",
"@teachntech00 New iPad Apps For #SpeechTherapy And Communication Are Showcased At The #SXSW Con
M) #iear #edchat #asd',
nan,
'#SXSW is just starting, #CTIA is around the corner and #googleio is only a hop skip and a jump
roid fan',
'Beautifully smart and simple idea RT @madebymany @thenextweb wrote about our #hollergram iPad a
bit.ly/ieaVOB'),
'Counting down the days to #sxsw plus strong Canadian dollar means stock up on Apple gear']
```

Tokenize the Words

```
In [20]: tokenz = word_tokenize('.'.join(str(v) for v in corpus))
```

```
In [21]: tokenz[:10]
```

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

Create Stopwords List

```
In [22]: stopwords_list = list(nlp.Defaults.stop_words)
len(nlp.Defaults.stop_words)
```

326

```
In [23]: stopwords_list
          another ,
          'already',
          'us',
          'thereafter',
          'until',
          ''re',
          'only',
          'few',
          'themselves',
          'hers',
          'n't',
          'll',
          'your',
          'also',
          'such',
          'nevertheless',
          'much',
          'be',
          'seeming',
          'meanwhile',
          'mine',
          're',
          'one',
          'whereafter',
          "m",
```

```
In [24]: stopwords_list.extend(string.punctuation)
```

```
In [25]: len(stopwords_list)
```

358

```
In [26]: stopwords_list.extend(stopwords.words('english'))
```

```
In [27]: len(stopwords_list)
```

537


```
In [28]: additional_punc = ['“', '”', '...', '""', "'", '`', 'https', 'rt', '\\.+']
stopword_list.extend(additional_punc)
stopword_list[-10:]
```

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

Remove stopwords and additional punctuation from the dat

```
In [29]: stopped_tokenz = [word.lower() for word in tokenz if word.lower() not in stop
```

```
In [30]: freq = FreqDist(stopped_tokenz)
freq.most_common(50)

[('sxsx', 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 UR misc

```
In [31]: 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 [32]: lemmatizer = WordNetLemmatizer()
```

```
In [33]: clean_stopped_tokenz = [word.lower() for word in stopped_tokenz if word not :  
clean_lemmatized_tokenz = [lemmatizer.lemmatize(word.lower()) for word in sto
```

```
In [34]: freq_clean_lemma = FreqDist(clean_lemmatized_tokenz)  
freq_lemma = freq_clean_lemma.most_common(5000)  
freq_lemma2 = freq_clean_lemma.most_common(25)
```

```
In [35]: total_word_count = len(clean_lemmatized_tokenz)
```

```
In [36]: lemma_word_count = sum(freq_clean_lemma.values()) # just a number
```

```
In [37]: 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 [38]: # from wordcloud import WordCloud

          # ## Inititalize 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))

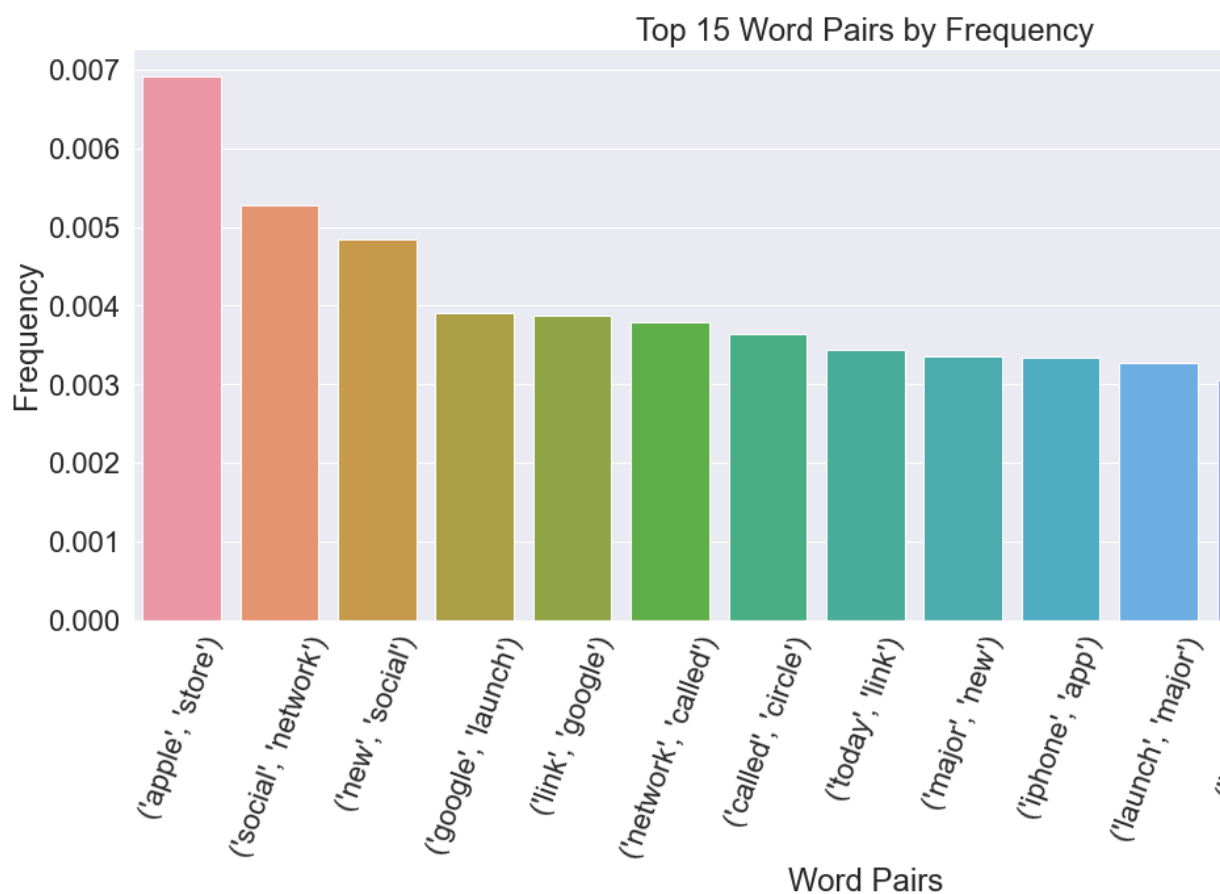
          # ## Plot with matplotlib
          # plt.figure(figsize = (12, 12), facecolor = None)
          # plt.imshow(wordcloud)
          # plt.axis('off')
```

```
In [39]: bigram_measures = nltk.collocations.BigramAssocMeasures()
          tweet_finder = nltk.BigramCollocationFinder.from_words(clean_lemmatized_token
          tweets_scored = tweet_finder.score_ngrams(bigram_measures.raw_freq)
```

```
In [47]: 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 [58]: 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 [59]: tweet_pmi_finder = nltk.BigramCollocationFinder.from_words(clean_lemmatized_tweets)
tweet_pmi_finder.apply_freq_filter(5)

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

In [62]:

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

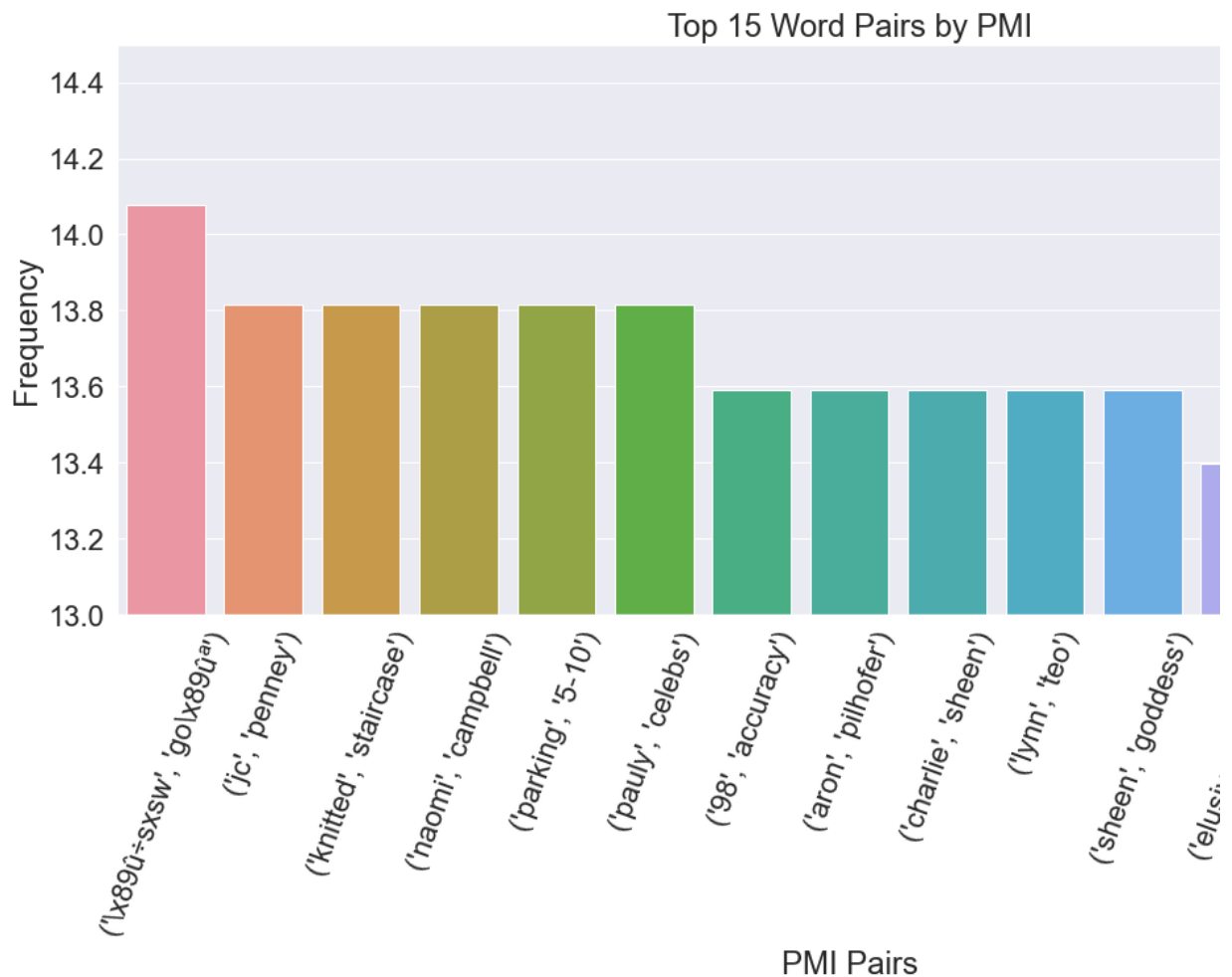
```
PMI_list
```

	Words	PMI
0	($\square \hat{u} + \text{sxsw}$, $\text{go} \square \hat{u}^a$)	14.076983
1	(jc, penney)	13.813948
2	(knitted, staircase)	13.813948
3	(naomi, campbell)	13.813948
4	(parking, 5-10)	13.813948
5	(pauly, celebs)	13.813948
6	(98, accuracy)	13.591556
7	(aron, pilhofer)	13.591556
8	(charlie, sheen)	13.591556
9	(lynn, teo)	13.591556
10	(sheen, goddess)	13.591556
11	(elusive, 'power)	13.398911
12	(zazzlsxsw, $\text{you} \square \hat{u}^a \text{ll}$)	13.398911
13	(cameron, sinclair)	13.398911
14	(sinclair, spearhead)	13.398911
15	(staircase, attendance)	13.398911

```

In [68]: fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=PMI_list.head(15)['Words'], y=PMI_list.head(15)['PMI'], palette=palette)
ax.set(xlabel="PMI Pairs",ylabel="Frequency")
plt.ylim([13,14.5])
plt.ticklabel_format(style='plain',axis='y')
plt.xticks(rotation=70)
plt.title('Top 15 Word Pairs by PMI')
plt.show()

```



In [69]:

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

	Tweet	Platform	Emotion	Uncertain
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	0
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive emotion	0
3	@sxsxw 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	0

```
In [70]: df1 = df1.drop(columns=['Uncertain', 'No Emotion'])
# Turn negative and positive columns into one column of just negatives and p
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
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	@sxsxw 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 [71]: from sklearn.utils import resample
```

```
In [72]: df_majority = df1.loc[df1['Positive_Bin']==1]
df_minority = df1.loc[df1['Positive_Bin']==0]
```

```
In [73]: df_minority.shape

(570, 4)
```

```
In [74]: df_majority.shape

(2978, 4)
```

```
In [75]: df_min_sample = resample(df_minority, replace=True, n_samples=1000, random_st
```

```
In [76]: df_maj_sample = resample(df_majority, replace=True, n_samples=2500, random_st
```

```
In [77]: df_upsampled = pd.concat([df_min_sample, df_maj_sample], axis=0)
df_upsampled.shape

(3500, 4)
```

```
In [78]: X, y = df_upsampled['Tweet'], df_upsampled['Positive_Bin']
```

Train/Test Split

```
In [79]: 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 [80]: scaler_object = MaxAbsScaler()
```

```
In [81]: 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 [82]: y_train.value_counts(0)
y_test.value_counts(1)

2020-12-17 12:53:52,995 : 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 [83]: from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.ensemble import RandomForestClassifier

tokenizer = nltk.TweetTokenizer(preserve_case=False)

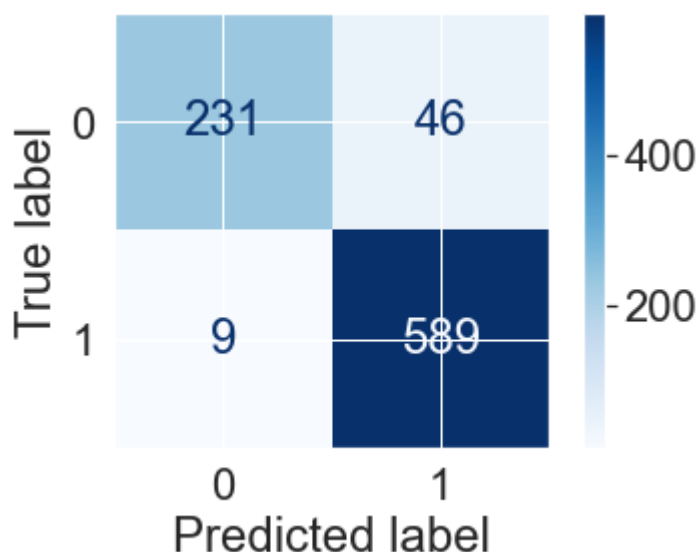
vectorizer = CountVectorizer(tokenizer=tokenizer.tokenize,
                             stop_words=stopword_list, decode_error='ignore')
```

```
In [84]: # for row in X_train:
#         for word in row:
#             Lemmatizer.Lemmatize(X_train[row][word])
# return X_train[word][row]
```

```
In [85]: X_train_count = vectorizer.fit_transform(X_train)
X_test_count = vectorizer.transform(X_test)
```

```
In [88]: evaluate_model(y_test, y_hat_test, X_test_count,clf=model) # 1 denotes Posit
```

	precision	recall	f1-score	support
0	0.96	0.83	0.89	277
1	0.93	0.98	0.96	598
accuracy			0.94	875
macro avg	0.95	0.91	0.92	875
weighted avg	0.94	0.94	0.94	875



```
In [89]: tf_idf_vectorizer = TfidfVectorizer(tokenizer=tokenizer.tokenize,
                                             stop_words=stopword_list,decode_error='i{
```

```
In [90]: 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: You should consider using 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,)

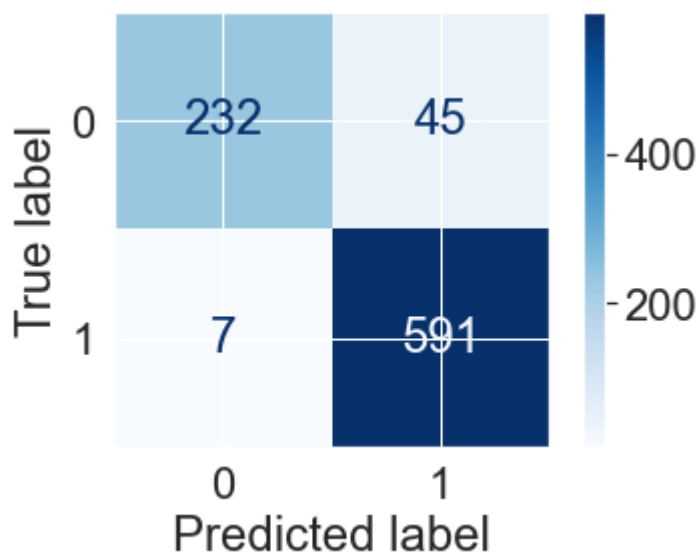
```
In [91]: from sklearn.ensemble import RandomForestClassifier
```

```
In [92]: ran_for = RandomForestClassifier(class_weight='balanced')
model_tf_idf = ran_for.fit(X_train_tf_idf,y_train)
```

```
In [93]: y_hat_tf_idf = model_tf_idf.predict(X_test_count)
```

```
In [94]: evaluate_model(y_test, y_hat_tf_idf, X_test_tf_idf,clf=model_tf_idf) # slight
```

	precision	recall	f1-score	support
0	0.90	0.64	0.75	277
1	0.85	0.97	0.91	598
accuracy			0.86	875
macro avg	0.88	0.80	0.83	875
weighted avg	0.87	0.86	0.86	875

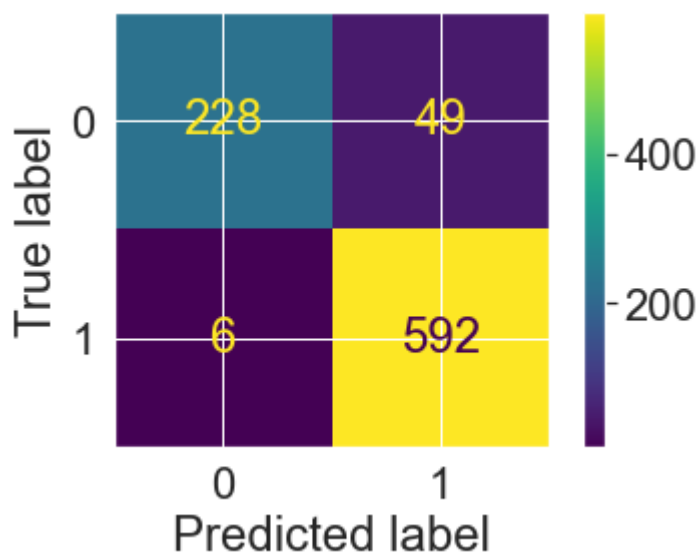


```
In [95]: ran_for = RandomForestClassifier()
ada_clf = AdaBoostClassifier()
gb_clf = GradientBoostingClassifier()

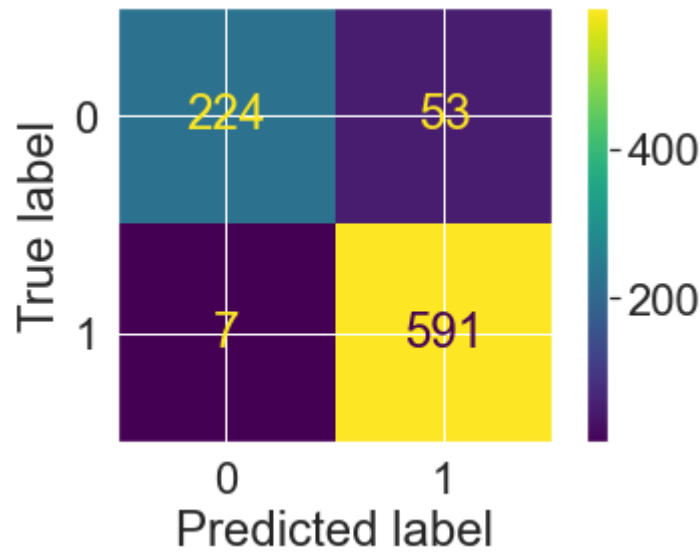
models = [ran_for, ada_clf, gb_clf]

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

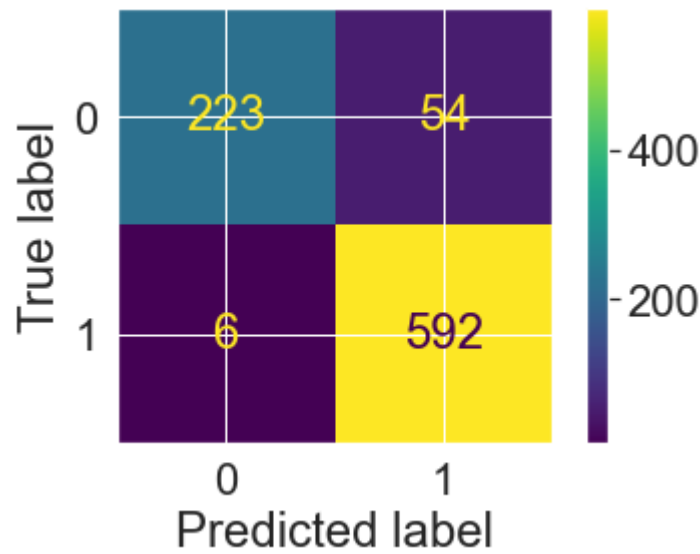
```
Accuracy Score: 0.9371428571428572
Precision Score: 0.9235569422776911
Recall Score: 0.9899665551839465
F1 Score: 0.9556093623890234
RandomForestClassifier() 0.9371428571428572
```



```
Accuracy Score: 0.9314285714285714
Precision Score: 0.9177018633540373
Recall Score: 0.9882943143812709
F1 Score: 0.9516908212560387
RandomForestClassifier() 0.9314285714285714
```



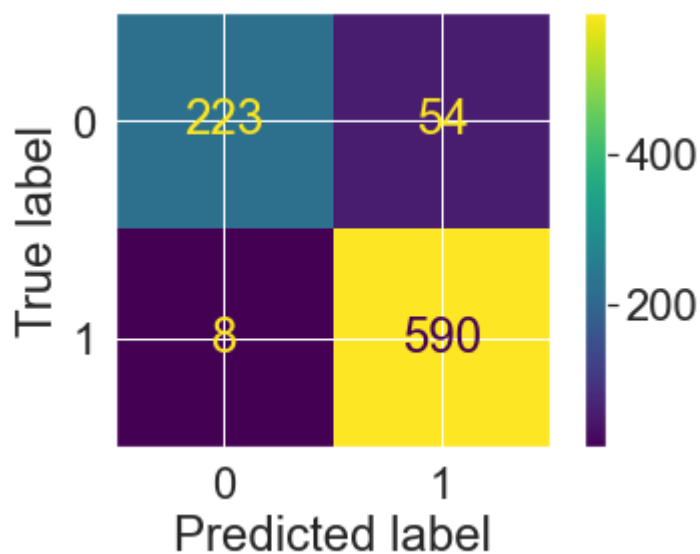
Accuracy Score: 0.9314285714285714
Precision Score: 0.9164086687306502
Recall Score: 0.9899665551839465
F1 Score: 0.9517684887459806
RandomForestClassifier() 0.9314285714285714



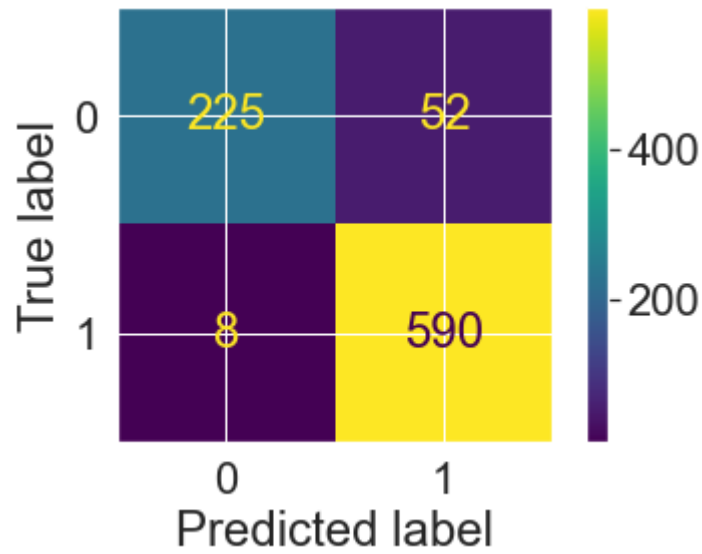
In [96]:

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

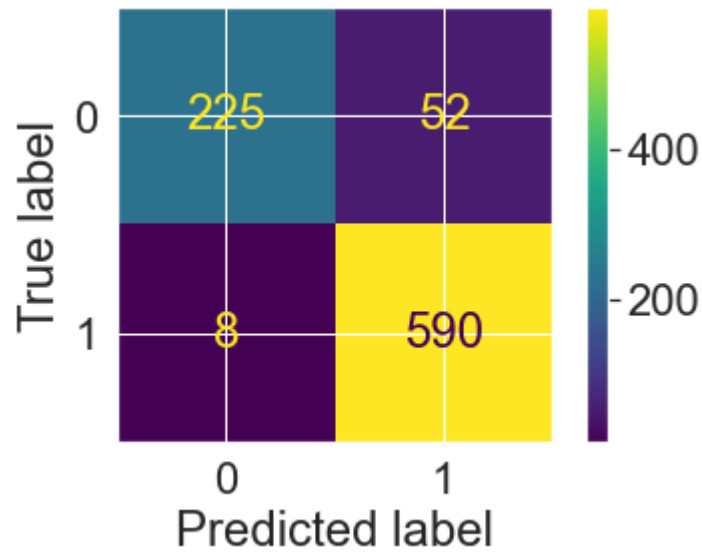
```
Accuracy Score: 0.9291428571428572  
Precision Score: 0.9161490683229814  
Recall Score: 0.9866220735785953  
F1 Score: 0.9500805152979066  
RandomForestClassifier() 0.9291428571428572
```



```
Accuracy Score: 0.9314285714285714  
Precision Score: 0.9190031152647975  
Recall Score: 0.9866220735785953  
F1 Score: 0.9516129032258065  
RandomForestClassifier() 0.9314285714285714
```



Accuracy Score: 0.9314285714285714
Precision Score: 0.9190031152647975
Recall Score: 0.9866220735785953
F1 Score: 0.9516129032258065
RandomForestClassifier() 0.9314285714285714



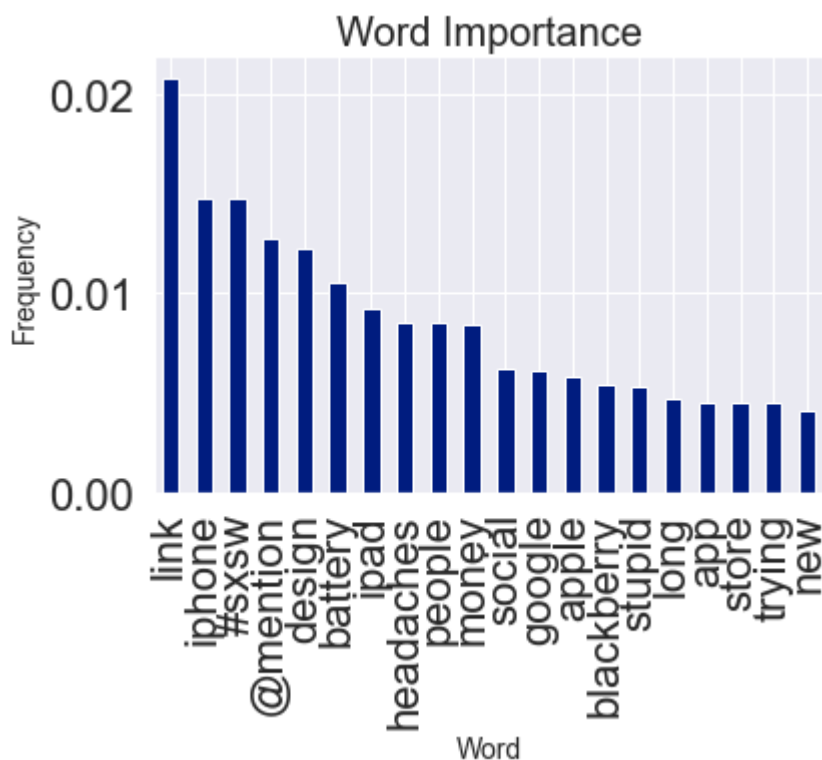
```
In [97]: tf_idf_vectorizer.get_feature_names()
```

```
'#15slides',  
'#310409h2011',  
'#4sq',  
'#911tweets',  
'#abacus',  
'#accessxsw',  
'#accordion',  
'#aclu',  
'#adam',  
'#addictedtotheinterwebs',  
'#adpeopleproblems',  
'#agchat',  
'#agileagency',  
'#agnerd',  
'#allhat3',  
'#alwayshavingtoplugin',  
'#amateurhour',  
'#android',  
"#android's",  
'#androidxsw',  
'#angrybirds',  
'#annoying',  
'#app',  
'#appcircuit',  
'#apple'
```

```
In [98]: importance = pd.Series(ran_for.feature_importances_, index=tf_idf_vectorizer.get_feature_names_out())
importance

##sxsx      1.617660e-06
#10         3.874117e-07
#106        0.000000e+00
#11ntc      4.550715e-04
#1406-08    3.449528e-09
...
üimute     1.380013e-05
ùò         2.104275e-04
ùòand      3.954520e-04
ùó         1.350616e-04
ùóthe     1.488186e-04
Length: 4295, dtype: float64
```

```
In [99]: importance.sort_values(ascending=False).head(20).plot(kind='bar')
plt.title('Word Importance', fontsize=20)
plt.ylabel(ylabel='Frequency', fontsize=14)
plt.xlabel(xlabel='Word', fontsize=14)
plt.show()
```



```
In [100]: vectorizer = CountVectorizer()
tf_transform = TfidfTransformer(use_idf=True)
```

```
In [101]: text_pipe = Pipeline(steps=[
            ('count_vectorizer',vectorizer),
            ('tf_transformer',tf_transform)])
```

```
In [102]: RandomForestClassifier(class_weight='balanced')

RandomForestClassifier(class_weight='balanced')
```

```
In [103]: full_pipe = Pipeline(steps=[
            ('text_pipe',text_pipe),
            ('clf',RandomForestClassifier(class_weight='balanced'))
        ])
```

```
In [104]: X_train_pipe = text_pipe.fit_transform(X_train)
```

```
In [105]: X_test_pipe = text_pipe.transform(X_test)
```

```
In [106]: X_train_pipe

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

```
In [107]: 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 [108]: ## Make and fit grid
          grid = GridSearchCV(full_pipe, params, cv=3)
          grid.fit(X_train, y_train)
          ## Display best params
          grid.best_params_

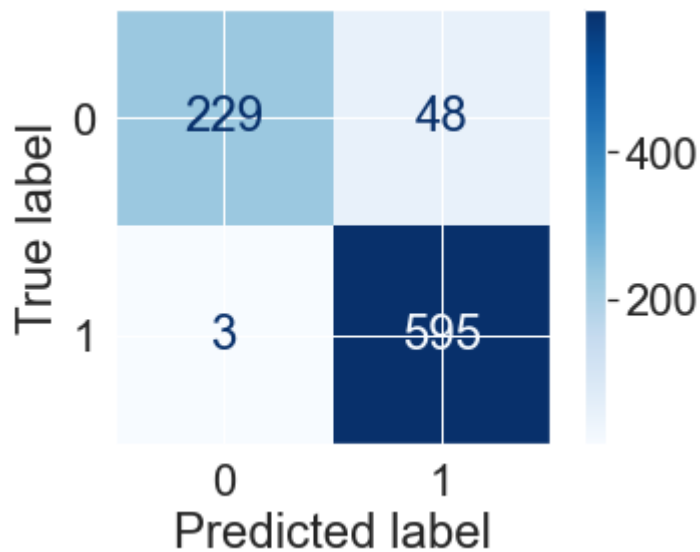
          'indeed',
          'than',
          'twelve',
          'm',
          'against',
          'doing',
          'yours',
          'everywhere',
          'whereas',
          'down',
          'further',
          'whole',
          'of',
          'whenever',
          'top',
          'anyone',
          'to',
          'together',

          'below',
          'out',
          'within',
          'here',
          'whose',
          'hundred'
```

```
In [109]: best_pipe = grid.best_estimator_
          y_hat_test = grid.predict(X_test)
```

```
In [110]: evaluate_model(y_test,y_hat_test,X_test,best_pipe)
```

	precision	recall	f1-score	support
0	0.99	0.83	0.90	277
1	0.93	0.99	0.96	598
accuracy			0.94	875
macro avg	0.96	0.91	0.93	875
weighted avg	0.94	0.94	0.94	875



```
In [111]: X_train_pipe.shape
```

```
(2625, 4256)
```

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

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

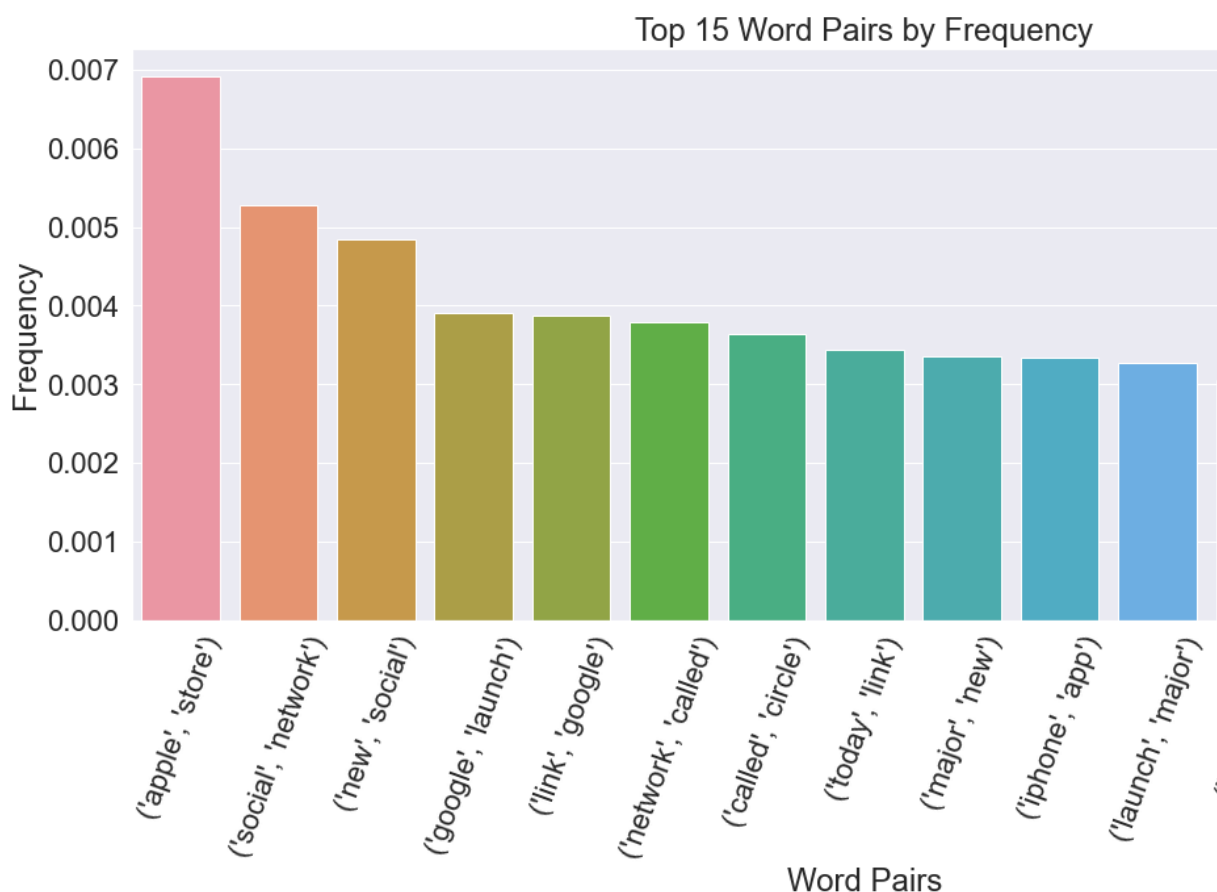
```
In [150]: bigram_measures = nltk.collocations.BigramAssocMeasures()
tweet_finder = nltk.BigramCollocationFinder.from_words(clean_lemmatized_token
tweets_scored = tweet_finder.score_ngrams(bigram_measures.raw_freq)
```

```
In [151]: 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	(âç, complete)	0.000012
42703	(âçwhat, tech)	0.000012
42704	(âè, android)	0.000012
42705	(âè, ubersoc)	0.000012
42706	(\u00b1g, wish)	0.000012

42707 rows x 2 columns


```
In [152]: fig_dims = (20,8)
fig, ax = plt.subplots(figsize=fig_dims)
sns.set(font_scale=2)
sns.set_style("darkgrid")
palette = sns.set_palette("dark")
ax = sns.barplot(x=bigram1.head(15)['Words'], y=bigram1.head(15)['Freq'], palette=palette)
ax.set(xlabel="Word Pairs",ylabel="Frequency")
plt.ticklabel_format(style='plain',axis='y')
plt.xticks(rotation=70)
plt.title('Top 15 Word Pairs by Frequency')
plt.show()
```



Deep NLP using Keras NN

```
In [116]: from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras import models, layers, optimizers
```

```
In [117]: model = 0
```

```
In [118]: y = np.asarray(df_upsampled['Positive_Bin']).astype('float32').reshape((-1,1))
X = np.asarray(df_upsampled['Tweet'])
```

```
In [119]: X.dtype
```

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

```
In [120]: tokenizer = Tokenizer()
tokenizer.fit_on_texts(list(X))
sequences = tokenizer.texts_to_sequences(X)
X = sequence.pad_sequences(sequences,maxlen=100)
```

```
In [121]: tokenizer.word_counts
          ('your', 100),
          ('we', 86),
          ('get', 199),
          ('it', 480),
          ('its', 58),
          ('big', 34),
          ('deal', 6),
          ('everybody', 3),
          ('has', 138),
          ('one', 149),
          ('overheard', 9),
          ('interactive', 34),
          ('quot', 667),
          ('arg', 4),
          ('i', 703),
          ('hate', 11),
          ('want', 64),
          ('my', 446),
          ('blackberry', 39),
          ('back', 48),
          ('shocked', 3),
          ('virtualwallet', 2),
          ('nfc', 2),
          ('iphone5', 2),
          ('bc', 2),
```

```
In [122]: vocab_size = len(tokenizer.word_counts)
seq_len = X.shape[1]
```

```
In [123]: vocab_size
          seq_len

          100
```

```
In [124]: print(type(X),X.shape)
          print(type(y),y.shape)

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

```
In [125]: X = np.asarray(X).astype('float32')
```

```
In [126]: print(type(X),X.shape)
          print(type(y),y.shape)

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

```
In [127]: # X_train, y_train, X_test, y_test = train_test_split(X, y, random_state=42,
```

```
In [128]: 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 [129]:

```
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		
=====		

In [130]:

```
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-130-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(
rbose, callbacks, validation_split, validation_data, shuffle, class_weight, sample_weight, initial
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, **kwargs)
```

```
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(
```

```
869         # This is the first call of __call__, so we have to initialize.
```

```
870         initializers = []
```

```
--> 871         self._initialize(args, kwargs, 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 _initia
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: d
*args, **kwargs))
```

```
727
```

```
~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\eager\function.py in _get_concre
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_def:
```

```
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_
nc, args, kwargs, signature, func_graph, autograph, autograph_options, add_control_dependencies, i
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
    632         xla_context.Exit()
    633     else:
--> 634         out = weak_wrapped_fn().__wrapped__(*args, **kwargs)
    635     return out
    636

~\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\framework\func_graph.py in wrap
    975     except Exception as e: # pylint:disable=broad-exception
    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\ti
return step_function(self, iterator)
C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\ti
outputs = model.distribute_strategy.run(run_step, args=(data,))
C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\distribute\dis
return self._extended.call_for_each_replica(fn, args=args, kwargs=kwargs)
C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\distribute\dis
eplica
return self._call_for_each_replica(fn, args, kwargs)
C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\distribute\dis
replica
return fn(*args, **kwargs)
C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\ti
outputs = model.train_step(data)
C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\ti
loss = self.compiled_loss(
C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\engine\cc
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
losses = call_fn(y_true, y_pred)
C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\losses.py
return ag_fn(y_true, y_pred, **self._fn_kwargs)
C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\util\dispatch.i
return target(*args, **kwargs)

```

```

C:\Users\josep\AppData\Roaming\Python\Python38\site-packages\tensorflow\python\keras\losses.py
    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
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 [153]: `from nltk import word_tokenize`

In [154]: `data = df_upsampled['Tweet'].map(word_tokenize)`

In [155]: `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, ooling, 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

```

In [156]:

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

2020-12-17 13:06:20,152 : INFO : collecting all words and their counts
2020-12-17 13:06:20,153 : INFO : PROGRESS: at sentence #0, processed 0 words, keeping 0 word types
2020-12-17 13:06:20,171 : INFO : collected 5920 word types from a corpus of 86715 raw words and 31
2020-12-17 13:06:20,172 : INFO : Loading a fresh vocabulary
2020-12-17 13:06:20,184 : INFO : effective_min_count=1 retains 5920 unique words (100% of original
2020-12-17 13:06:20,185 : INFO : effective_min_count=1 leaves 86715 word corpus (100% of original
2020-12-17 13:06:20,204 : INFO : deleting the raw counts dictionary of 5920 items
2020-12-17 13:06:20,207 : INFO : sample=0.001 downsamples 52 most-common words
2020-12-17 13:06:20,208 : INFO : downsampling leaves estimated 56808 word corpus (65.5% of prior)
2020-12-17 13:06:20,219 : INFO : estimated required memory for 5920 words and 100 dimensions: 7696
2020-12-17 13:06:20,220 : INFO : resetting layer weights
2020-12-17 13:06:21,308 : INFO : training model with 4 workers on 5920 vocabulary and 100 features
active=5 window=5
2020-12-17 13:06:21,345 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,347 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,351 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,352 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,354 : INFO : EPOCH - 1 : training on 86715 raw words (56795 effective words) t
s
2020-12-17 13:06:21,397 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,402 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,404 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,405 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,407 : INFO : EPOCH - 2 : training on 86715 raw words (56775 effective words) t
s
2020-12-17 13:06:21,450 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,458 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,459 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,460 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,460 : INFO : EPOCH - 3 : training on 86715 raw words (56788 effective words) t
s
2020-12-17 13:06:21,504 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,508 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,509 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,510 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,510 : INFO : EPOCH - 4 : training on 86715 raw words (56940 effective words) t
s
2020-12-17 13:06:21,553 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,559 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,561 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,562 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,563 : INFO : EPOCH - 5 : training on 86715 raw words (56768 effective words) t
s
2020-12-17 13:06:21,563 : INFO : training on a 433575 raw words (284066 effective words) took 0.3s
```


In [157]:

```
model_W2V.train(data,total_examples=model_W2V.corpus_count, epochs=10)
```

```
2020-12-17 13:06:21,569 : WARNING : Effective 'alpha' higher than previous training cycles
2020-12-17 13:06:21,570 : INFO : training model with 4 workers on 5920 vocabulary and 100 features
active=5 window=5
2020-12-17 13:06:21,620 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,626 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,629 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,630 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,631 : INFO : EPOCH - 1 : training on 86715 raw words (56706 effective words) 1
s
2020-12-17 13:06:21,672 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,678 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,679 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,681 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,682 : INFO : EPOCH - 2 : training on 86715 raw words (56852 effective words) 1
s
2020-12-17 13:06:21,724 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,727 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,730 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,732 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,733 : INFO : EPOCH - 3 : training on 86715 raw words (56891 effective words) 1
s
2020-12-17 13:06:21,769 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,772 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,777 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,778 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,778 : INFO : EPOCH - 4 : training on 86715 raw words (56852 effective words) 1
s
2020-12-17 13:06:21,819 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,826 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,828 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,830 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,831 : INFO : EPOCH - 5 : training on 86715 raw words (56672 effective words) 1
s
2020-12-17 13:06:21,876 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,882 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,883 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,884 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,885 : INFO : EPOCH - 6 : training on 86715 raw words (56801 effective words) 1
s
2020-12-17 13:06:21,921 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,929 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,931 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,932 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,932 : INFO : EPOCH - 7 : training on 86715 raw words (56838 effective words) 1
s
2020-12-17 13:06:21,969 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:21,975 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:21,978 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:21,979 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:21,979 : INFO : EPOCH - 8 : training on 86715 raw words (56914 effective words) 1
s
2020-12-17 13:06:22,019 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:22,020 : INFO : worker thread finished; awaiting finish of 2 more threads
```

```
2020-12-17 13:06:22,024 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:22,026 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:22,028 : INFO : EPOCH - 9 : training on 86715 raw words (56831 effective words) i
s
2020-12-17 13:06:22,066 : INFO : worker thread finished; awaiting finish of 3 more threads
2020-12-17 13:06:22,074 : INFO : worker thread finished; awaiting finish of 2 more threads
2020-12-17 13:06:22,075 : INFO : worker thread finished; awaiting finish of 1 more threads
2020-12-17 13:06:22,076 : INFO : worker thread finished; awaiting finish of 0 more threads
2020-12-17 13:06:22,076 : INFO : EPOCH - 10 : training on 86715 raw words (56778 effective words)
s/s
2020-12-17 13:06:22,077 : INFO : training on a 867150 raw words (568135 effective words) took 0.5:

(568135, 867150)
```

```
In [158]: wv = model_W2V.wv
```

```
In [159]: wv.most_similar(positive='good')
```

```
2020-12-17 13:06:22,089 : INFO : precomputing L2-norms of word weight vectors
```

```
[('scan', 0.9812557697296143),
 ('vCards', 0.9642086625099182),
 ('They', 0.9605367183685303),
 ('longer', 0.9592171311378479),
 ('festival', 0.9579607844352722),
 ('behave', 0.9553024768829346),
 ('Simple', 0.9551275968551636),
 ('ipads', 0.9546160697937012),
 ('like', 0.9534077644348145),
 ('dude', 0.9532365798950195)]
```

In [160]:

wv['help']

```
array([ 0.26340348, -0.40018958,  0.06040369, -0.19309096, -0.25014228,
        0.11403151,  0.13322787, -0.1535964 ,  0.04666793,  0.22953562,
        0.32465532,  0.15474607, -0.00531764, -0.2743694 , -0.11493958,
        0.11698529,  0.02690452, -0.1391758 , -0.05665627,  0.17294443,
        0.14213072, -0.09188478,  0.00366103,  0.3397651 ,  0.11656451,
        0.17908143, -0.11154556,  0.26916105, -0.11514664, -0.10411742,
        -0.289618 , -0.0189602 ,  0.02211949, -0.16155374, -0.11159767,
        -0.06790099,  0.21793206, -0.11909045,  0.05444815,  0.01792222,
        -0.06722239,  0.02029637,  0.06175743,  0.21486077,  0.31147718,
        0.37647897, -0.30715406,  0.18051036, -0.07471982, -0.14325368,
        0.2517508 ,  0.16290008, -0.23652588,  0.01067481,  0.11774933,
        0.31457067,  0.09200221,  0.07123999,  0.08954549, -0.06245711,
        0.0696683 ,  0.09090552,  0.24627443, -0.09342247, -0.14598 ,
        0.22235683,  0.04680994,  0.11824694, -0.7169352 , -0.26919207,
        0.05049581,  0.00485155, -0.15389952, -0.11464192, -0.14360659,
        -0.22030841,  0.1552505 ,  0.17743054,  0.0372517 ,  0.12613313,
        0.02473311,  0.25920096, -0.10323504, -0.1841724 ,  0.14494286,
        0.15176162,  0.06931927,  0.07450345, -0.25900677, -0.2686803 ,
        0.00593456, -0.10767873, -0.09033866,  0.14090653,  0.02257674,
        0.10058074, -0.13294472,  0.2173957 ,  0.00932731,  0.08059204],
      dtype=float32)
```

In [161]:

wv.vectors

```
array([[ 7.5755513e-01,  1.7843774e-02,  3.1630000e-01, ...,
        5.7815367e-01,  5.8852291e-01, -1.6978522e-01],
       [ 7.1542639e-01, -7.6082975e-01,  5.3152555e-01, ...,
        5.2534026e-01, -2.0729542e-02,  3.2103598e-01],
       [ 1.6311506e+00, -1.8997508e+00, -2.5010246e-01, ...,
        2.1062391e-01,  3.3783191e-01, -6.9523197e-01],
       ...,
       [ 4.3708596e-02, -4.8188835e-02,  1.0424881e-02, ...,
        2.1558555e-02,  2.4326772e-03,  4.7256039e-03],
       [-5.6209791e-02, -2.0330665e-03, -8.7956796e-05, ...,
        -5.8318313e-02,  2.5095209e-02, -1.1613940e-02],
       [ 4.5090374e-03, -2.8578965e-02,  1.8759282e-02, ...,
        7.0168213e-03, -2.3256097e-02, -9.8043354e-04]], dtype=float32)
```

```
In [162]: wv.most_similar(positive=['apple','google'], negative = ['man'])

[('sprinkle', 0.7258357405662537),
 ('smartphones', 0.6931923627853394),
 ('Stacks', 0.6855062246322632),
 ('phenomenal', 0.6852372884750366),
 ('mylunch', 0.6822008490562439),
 ('googledoodle', 0.6816586852073669),
 ('technology', 0.6721771955490112),
 ('LatinasInTech\x890\x9d', 0.6605688333511353),
 ('Geek', 0.6575497984886169),
 ('nerdheaven', 0.6560829281806946)]
```

```
In [ ]:
```