Importing necessary libraries

In [99]:

```
!pip install gensim
!pip install worldcloud
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: gensim in /home/aditi/.local/lib/python3.8/site-packages (3.8.3)
Requirement already satisfied: numpy>=1.11.3 in /home/aditi/.local/lib/python3.8/site-packages
(from gensim) (1.19.1)
Requirement already satisfied: scipy>=0.18.1 in /home/aditi/.local/lib/python3.8/site-packages
(from gensim) (1.5.2)
Requirement already satisfied: six>=1.5.0 in /usr/lib/python3/dist-packages (from gensim) (1.14.0)
Requirement already satisfied: smart-open>=1.8.1 in /home/aditi/.local/lib/python3.8/site-packages
(from gensim) (4.0.0)
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (from smart-open>=1.8.1-
>gensim) (2.22.0)
WARNING: You are using pip version 20.2.2; however, version 20.2.4 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
Defaulting to user installation because normal site-packages is not writeable
ERROR: Could not find a version that satisfies the requirement worldcloud (from versions: none)
ERROR: No matching distribution found for worldcloud
WARNING: You are using pip version 20.2.2; however, version 20.2.4 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
```

In [100]:

```
!pip install nltk
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: nltk in /home/aditi/.local/lib/python3.8/site-packages (3.5)
Requirement already satisfied: regex in /home/aditi/.local/lib/python3.8/site-packages (from nltk)
(2020.7.14)
Requirement already satisfied: tqdm in /home/aditi/.local/lib/python3.8/site-packages (from nltk)
(4.49.0)
Requirement already satisfied: click in /home/aditi/.local/lib/python3.8/site-packages (from nltk)
(7.1.2)
Requirement already satisfied: joblib in /home/aditi/.local/lib/python3.8/site-packages (from
nltk) (0.16.0)
WARNING: You are using pip version 20.2.2; however, version 20.2.4 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
```

In [101]:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
```

In [104]:

```
train = pd.read_csv('train_tweet.csv')
test = pd.read_csv('test_tweets.csv')
```

In [105]:

```
print(train.shape)
print(test.shape)
```

```
(31962, 3)
(17197, 2)
```

In [106]:

```
train.head()
```

Out[106]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

In [107]:

```
test.head()
```

Out[107]:

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedic...
1	31964	@user #white #supremacists want everyone to s...
2	31965	safe ways to heal your #acne!! #altwaystohe...
3	31966	is the hp and the cursed child book up for res...
4	31967	3rd #bihday to my amazing, hilarious #nephew...

In [108]:

```
train.isnull().any()
test.isnull().any()
```

Out[108]:

```
id       False
tweet    False
dtype: bool
```

Checking out the negative comments from train dataset

In [109]:

```
train[train['label']==0].head(5)
```

Out[109]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

Checking out the positive comments from tarin dataset

In [110]:

```
train[train['label']==1].head(5)
```

Out[110]:

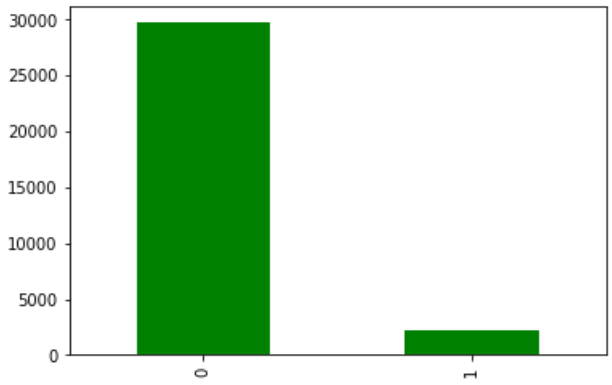
	id	label	tweet
13	14	1	@user #cnn calls #michigan middle school 'buil...
14	15	1	no comment! in #australia #opkillingbay #se...
17	18	1	retweet if you agree!
23	24	1	@user @user lumpy says i am a . prove it lumpy.
34	35	1	it's unbelievable that in the 21st century we'...

In [111]:

```
train['label'].value_counts().plot.bar(color = 'green', figsize = (6, 4))
```

Out[111]:

<AxesSubplot:>



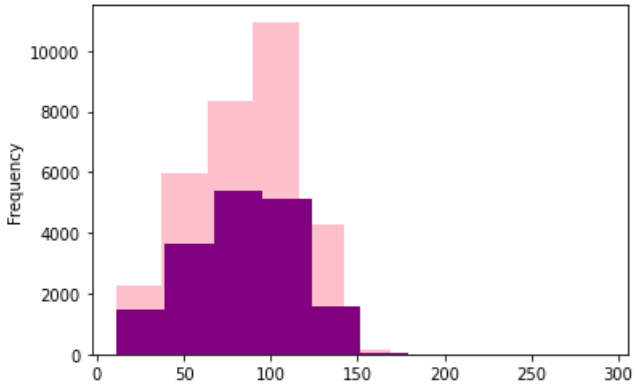
Observations:

1. Negative tweets exceed positive tweets largely in number

Checkng the distribution of tweets in the data

In [112]:

```
length_train = train['tweet'].str.len().plot.hist(color = 'pink', figsize = (6, 4))
length_test = test['tweet'].str.len().plot.hist(color = 'purple', figsize = (6, 4))
```



Adding column to represent length of the tweet

In [113]:

```
train['len'] = train['tweet'].str.len()
test['len'] = test['tweet'].str.len()
```

In [114]:

```
train.head(10)
```

Out[114]:

	id	label	tweet	len
0	1	0	@user when a father is dysfunctional and is s...	102
1	2	0	@user @user thanks for #lyft credit i can't us...	122
2	3	0	bihday your majesty	21
3	4	0	#model i love u take with u all the time in ...	86
4	5	0	factsguide: society now #motivation	39
5	6	0	[2/2] huge fan fare and big talking before the...	116
6	7	0	@user camping tomorrow @user @user @user @use...	74
7	8	0	the next school year is the year for exams.ð...	143
8	9	0	we won!!! love the land!!! #allin #cavs #champ...	87
9	10	0	@user @user welcome here ! i'm it's so #gr...	50

Numeric data analysis

In [115]:

```
train.groupby('label').describe()
```

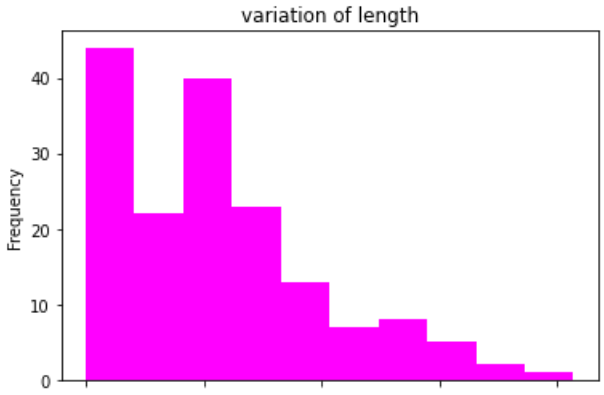
Out[115]:

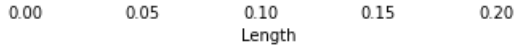
	id								len					
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%
label														
0	29720.0	15974.454441	9223.783469	1.0	7981.75	15971.5	23965.25	31962.0	29720.0	84.328634	29.566484	11.0	62.0	88.0
1	2242.0	16074.896075	9267.955758	14.0	8075.25	16095.0	24022.00	31961.0	2242.0	90.187779	27.375502	12.0	69.0	96.0

Observing the length of tweets generally used

In [116]:

```
train.groupby('len').mean()['label'].plot.hist(color = 'magenta', figsize = (6, 4),)
plt.title('variation of length')
plt.xlabel('Length')
plt.show()
```





CountVectorization

returns each unique word as a feature with the count of number of times that word occurs.

In [117]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

In [118]:

```
cv = CountVectorizer(stop_words = 'english')
words = cv.fit_transform(train.tweet)
```

In [119]:

```
sum_words = words.sum(axis=0)
```

In [120]:

```
words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)
```

In [121]:

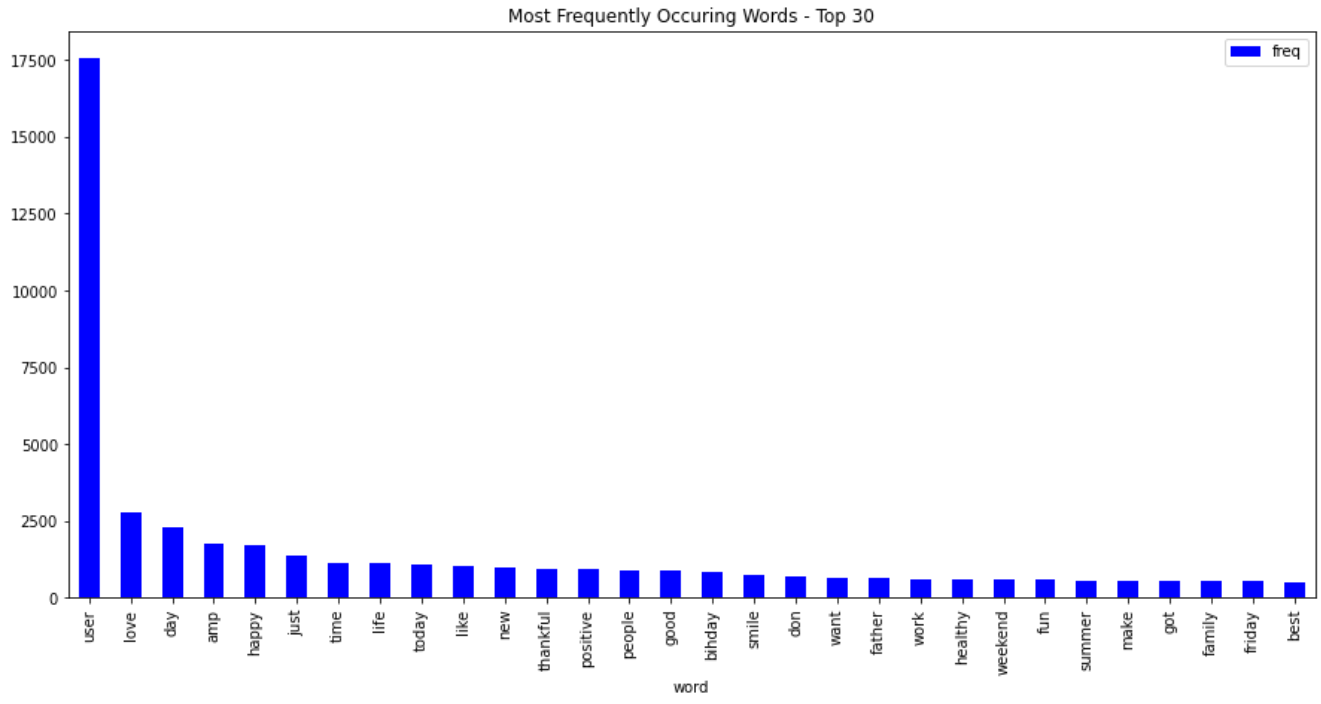
```
frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])
```

In [122]:

```
frequency.head(30).plot(x='word', y='freq', kind='bar', figsize=(15, 7), color = 'blue')
plt.title("Most Frequently Occuring Words - Top 30")
```

Out[122]:

```
Text(0.5, 1.0, 'Most Frequently Occuring Words - Top 30')
```



WordCloud

In [123]:

```
from wordcloud import WordCloud
wordcloud = WordCloud(background_color = 'white', width = 1000, height =
1000).generate_from_frequencies(dict(words_freq))
```

In [124]:

```
plt.figure(figsize=(10,8))
plt.imshow(wordcloud)
plt.title("WordCloud - Vocabulary from Reviews", fontsize = 22)
```

Out[124]:

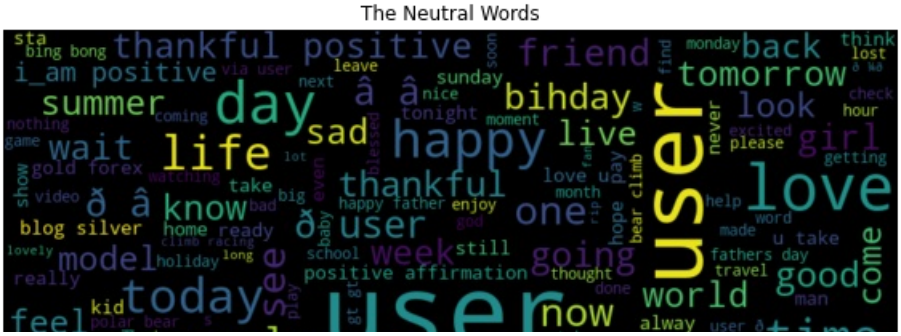
```
Text(0.5, 1.0, 'WordCloud - Vocabulary from Reviews')
```

In [125]:

```
normal_words =' '.join([text for text in train['tweet'][train['label'] == 0]])
```

In [126]:

```
wordcloud = WordCloud(width=800, height=500, random_state = 0, max_font_size = 110).generate(normal
_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('The Neutral Words')
plt.show()
```

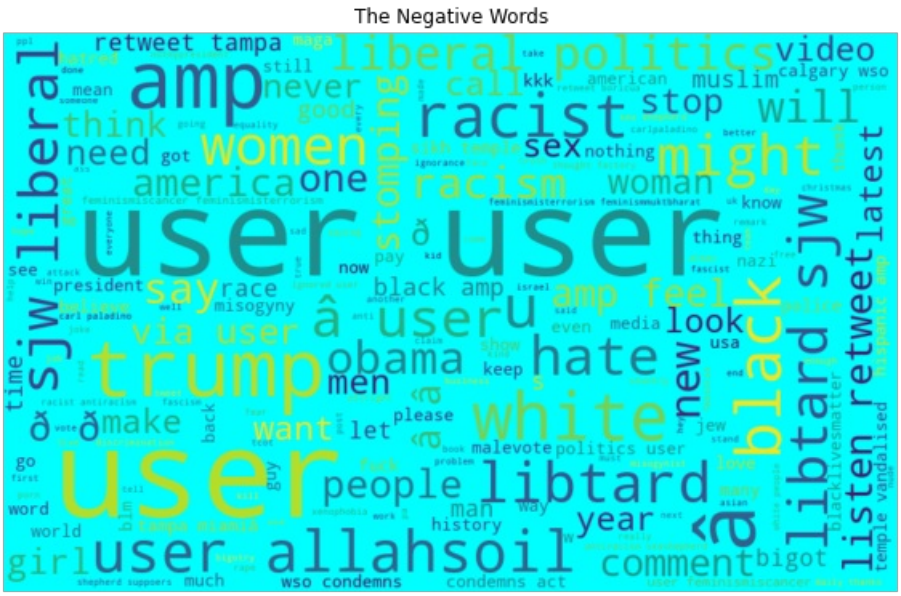


In [127]:

```
negative_words =' '.join([text for text in train['tweet'][train['label'] == 1]])
```

In [128]:

```
wordcloud = WordCloud(background_color = 'cyan', width=800, height=500, random_state = 0, max_font_
size = 110).generate(negative_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('The Negative Words')
plt.show()
```



Collecting hashtags

In [129]:

```
import re
def hashtag_extract(x):
    hashtags = []

    for i in x:
        ht = re.findall(r"#(\w+)", i)
        hashtags.append(ht)

    return hashtags
```

extracting hashtags from non racist/sexist tweets

In [130]:

```
HT_regular = hashtag_extract(train['tweet'][train['label'] == 0])
```

extracting hashtags from sexist/racist tweets

In [131]:

```
HT_negative = hashtag_extract(train['tweet'][train['label'] == 1])
```

unnesting list

In [132]:

```
HT_regular = sum(HT_regular,[])
HT_negative = sum(HT_negative,[])
```

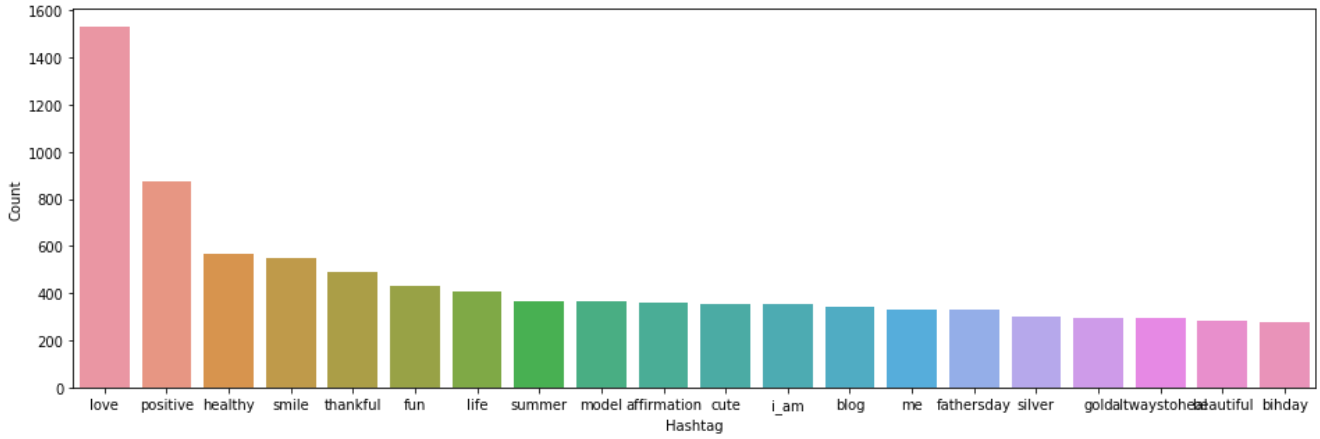
In [133]:

```
import nltk
a = nltk.FreqDist(HT_regular)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})
```

selecting top 20 most frequent hashtags

In [134]:

```
d = d.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```

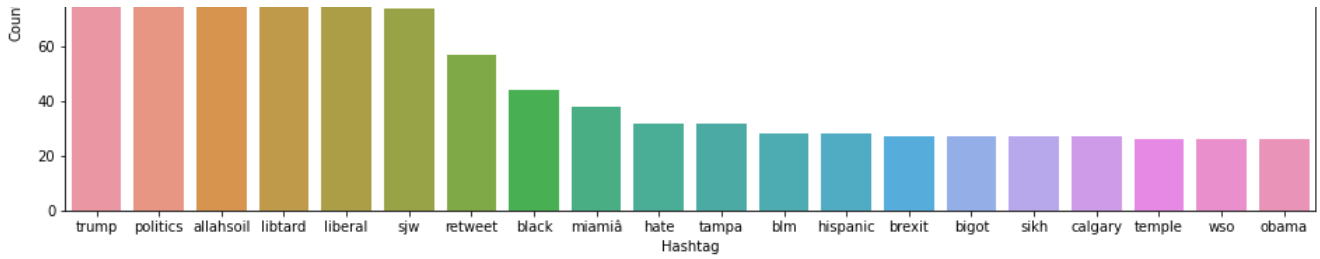


In [135]:

```
a = nltk.FreqDist(HT_negative)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})
```

In [136]:

```
d = d.nlargest(columns="Count", n = 20)
plt.figure(figsize=(16,5))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```



tokenizing te words present in training set

In [137]:

```
tokenized_tweet = train['tweet'].apply(lambda x: x.split())
```

importing gensim

In [138]:

```
import gensim
```

creating a word to vector model

In [139]:

```
model_w2v = gensim.models.Word2Vec(
            tokenized_tweet,
            size=200, # desired no. of features/independent variables
            window=5, # context window size
            min_count=2,
            sg = 1, # 1 for skip-gram model
            hs = 0,
            negative = 10, # for negative sampling
            workers= 2, # no.of cores
            seed = 34)
```

In [140]:

```
model_w2v.train(tokenized_tweet, total_examples= len(train['tweet']), epochs=20)
```

Out[140]:

```
(6109793, 8411580)
```

In [141]:

```
model_w2v.wv.most_similar(positive = "dinner")
```

Out[141]:

```
[('spaghetti', 0.691346287727356),
 ('#prosecco', 0.6418297290802002),
 ('#wanderlust', 0.617306113243103),
 ('galway', 0.602875828742981),
 ('#restaurant', 0.6008146405220032),
 ('#boardgames', 0.600536584854126),
 ('coaching', 0.5991450548171997),
 ('podium', 0.598253607749939),
 ('willow', 0.5955026149749756),
 ('fluffy', 0.5953262448310852)]
```

In [142]:

```
model_w2v.wv.most_similar(positive = "cancer")
```

Out[142]:

```
[('champion,', 0.7275875806808472),
 ('law.', 0.7197955250740051),
 ('targeted', 0.71098393201828),
 ('spots.', 0.7065681219100952),
 ('level.', 0.7059565782546997),
 ('ways.', 0.7006312608718872),
 ('politicizing', 0.70036381483078),
 ('ownership', 0.6973391771316528),
 ('aol', 0.6951942443847656),
 ('professionals', 0.69349205493927)]
```

In [143]:

```
model_w2v.wv.most_similar(positive = "apple")
```

Out[143]:

```
[('"mytraining"', 0.7102435827255249),
 ('mytraining', 0.7096908092498779),
 ('training"', 0.6898198127746582),
 ('app,', 0.6426267623901367),
 ('app', 0.6141518950462341),
 ('ta', 0.6078773140907288),
 ('"my', 0.6032952070236206),
 ('humans.', 0.5723394155502319),
 ("domino's", 0.5713151693344116),
 ('heroku', 0.5712091326713562)]
```

In [144]:

```
model_w2v.wv.most_similar(negative = "hate")
```

Out[144]:

```
[('#apple', -0.019845834001898766),
 ('#games', -0.02145865373313427),
 ('eyes', -0.047583505511283875),
 ('hands', -0.04793839901685715),
 ('stas', -0.04845249652862549),
 ('#fundraising', -0.05218462273478508),
 ('#yay', -0.05983911082148552),
 ('#hype', -0.062245190143585205),
 ('â\x80¦', -0.06898031383752823),
 ('season', -0.0690961480140686)]
```

In [145]:

```
from tqdm import tqdm
tqdm.pandas(desc="progress-bar")
from gensim.models.doc2vec import LabeledSentence
```

```
/home/aditi/.local/lib/python3.8/site-packages/tqdm/std.py:670: FutureWarning: The Panel class is
removed from pandas. Accessing it from the top-level namespace will also be removed in the next ve
rsion
  from pandas import Panel
```

In [146]:

```
def add_label(twt):
    output = []
    for i, s in zip(twt.index, twt):
        output.append(LabeledSentence(s, ["tweet_" + str(i)]))
    return output
```

label all tweets

In [147]:

```
labeled_tweets = add_label(tokenized_tweet)
```

```
<ipython-input-146-868d96c8c1ce>:4: DeprecationWarning: Call to deprecated `LabeledSentence`
(Class will be removed in 4.0.0, use TaggedDocument instead).
  output.append(LabeledSentence(s, ["tweet_" + str(i)]))
```

In [148]:

```
labeled_tweets[:6]
```

Out[148]:

```
[LabeledSentence(words=['@user', 'when', 'a', 'father', 'is', 'dysfunctional', 'and', 'is', 'so',
'selfish', 'he', 'drags', 'his', 'kids', 'into', 'his', 'dysfunction.', '#run'], tags=
['tweet_0']),
 LabeledSentence(words=['@user', '@user', 'thanks', 'for', '#lyft', 'credit', 'i', "can't", 'use',
'cause', 'they', "don't", 'offer', 'wheelchair', 'vans', 'in', 'pdx.', '#disapointed',
'#getthanked'], tags=['tweet_1']),
 LabeledSentence(words=['bihday', 'your', 'majesty'], tags=['tweet_2']),
 LabeledSentence(words=['#model', 'i', 'love', 'u', 'take', 'with', 'u', 'all', 'the', 'time', 'in
', 'urð\x9f\x93±!!!', 'ð\x9f\x98\x99ð\x9f\x98\x8eð\x9f\x91\x84ð\x9f\x91',
'ð\x9f\x92¦ð\x9f\x92¦ð\x9f\x92¦'], tags=['tweet_3']),
 LabeledSentence(words=['factsguide:', 'society', 'now', '#motivation'], tags=['tweet_4']),
 LabeledSentence(words=['[2/2]', 'huge', 'fan', 'fare', 'and', 'big', 'talking', 'before', 'they',
'leave.', 'chaos', 'and', 'pay', 'disputes', 'when', 'they', 'get', 'there.', '#allshowandnogo'],
tags=['tweet_5'])]
```

removing unwanted patterns from data

In [149]:

```
import re
import nltk
```

In [150]:

```
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to /home/aditi/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [151]:

```
train_corpus = []
```

In [152]:

```
for i in range(0, 31962):
  review = re.sub('[^a-zA-Z]', ' ', train['tweet'][i])
  review = review.lower()
  review = review.split()
```

In [153]:

```
ps = PorterStemmer()
```

Stemming

In [154]:

```
review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
```

Joining them back with space

In [155]:

```
review = ' '.join(review)
train_corpus.append(review)
```

In [156]:

```
test_corpus = []
```

In [157]:

```
for i in range(0, 17197):
  review = re.sub('[^a-zA-Z]', ' ', test['tweet'][i])
  review = review.lower()
  review = review.split()
```

In [158]:

```
ps = PorterStemmer()
```

stemming

In [159]:

```
review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
```

Joining them back with spaces

In [160]:

```
review = ' '.join(review)
test_corpus.append(review)
```

Creating bag of words

In [170]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

In [171]:

```
cv = CountVectorizer(max_features = 2500)
x = cv.fit_transform(train_corpus).toarray()
y = train.iloc[:, 1]
```

In [172]:

```
print(x.shape)
print(y.shape)
```

```
(1, 3)
(31962,)
```

In [164]:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features = 2500)
x_test = cv.fit_transform(test_corpus).toarray()

print(x_test.shape)
```

```
(1, 7)
```

Splitting training data into train and valid sets

In [168]:

```
from sklearn.model_selection import train_test_split
```

In [ ]:

```

```

In [ ]:

```

```

```
model = RandomForestClassifier()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("F1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

```
/usr/lib/python3/dist-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarnir
  from numpy.core.umath_tests import inner1d
Training Accuracy : 0.9944933461265696
Validation Accuracy : 0.951445376048054
F1 score : 0.6008230452674896
[[7311  121]
 [ 267  292]]
```

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_valid)

print("Training Accuracy :", model.score(x_train, y_train))
print("Validation Accuracy :", model.score(x_valid, y_valid))

# calculating the f1 score for the validation set
print("f1 score :", f1_score(y_valid, y_pred))

# confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print(cm)
```

```
Training Accuracy : 0.984773267698469
Validation Accuracy : 0.9410586910274058
f1 score : 0.5915004336513443
[[7179  253]
 [ 218  341]]
```