tweets-sentimental

April 2, 2024

1 Project Contributors

1.1 Mohit Kumar (21051065)

1.2 Yashdeep (21051103)

1.3 Satyajeet Behera (2105923)

1.4 Shubh Mittal (2105740)

1.5 Tanisha Sarkar (2105762)

1.6 Samriddhi Singh (2105742)

1.7 Hrishikesh Hazarika (21051136)

1.8 Shashank Deepak (2105827)

2 Submitted To

2.1 Mr. A. Ranjith

[ ]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
```

[26]:
```
train = pd.read_csv('/kaggle/input/tweets-dataset/train_tweet.csv')
test = pd.read_csv('/kaggle/input/tweets-dataset/test_tweets.csv')

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

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

[27]:
```
train.head()
```

[27]:
```
   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
```

[28]:
```
test.head()
```

[28]:
```
      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…
```

[29]:
```
train.isnull().any()
test.isnull().any()
```

[29]:
```
id       False
tweet    False
dtype: bool
```

[30]:
```
# checking out the negative comments from the train set

train[train['label'] == 0].head(10)
```

[30]:
```
   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
5   6      0  [2/2] huge fan fare and big talking before the…
6   7      0   @user camping tomorrow @user @user @user @use…
7   8      0  the next school year is the year for exams.ð …
8   9      0  we won!!! love the land!!! #allin #cavs #champ…
9  10      0   @user @user welcome here !  i'm   it's so #gr…
```

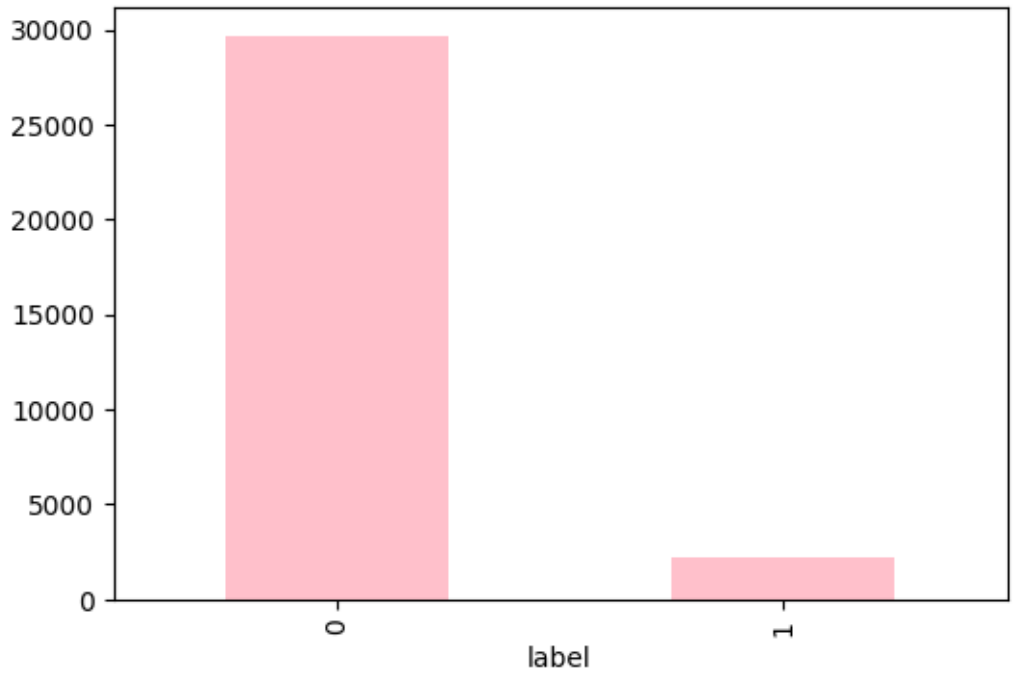
[31]:
```
# checking out the postive comments from the train set

train[train['label'] == 1].head(10)
```

[31]:
```
      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!
```

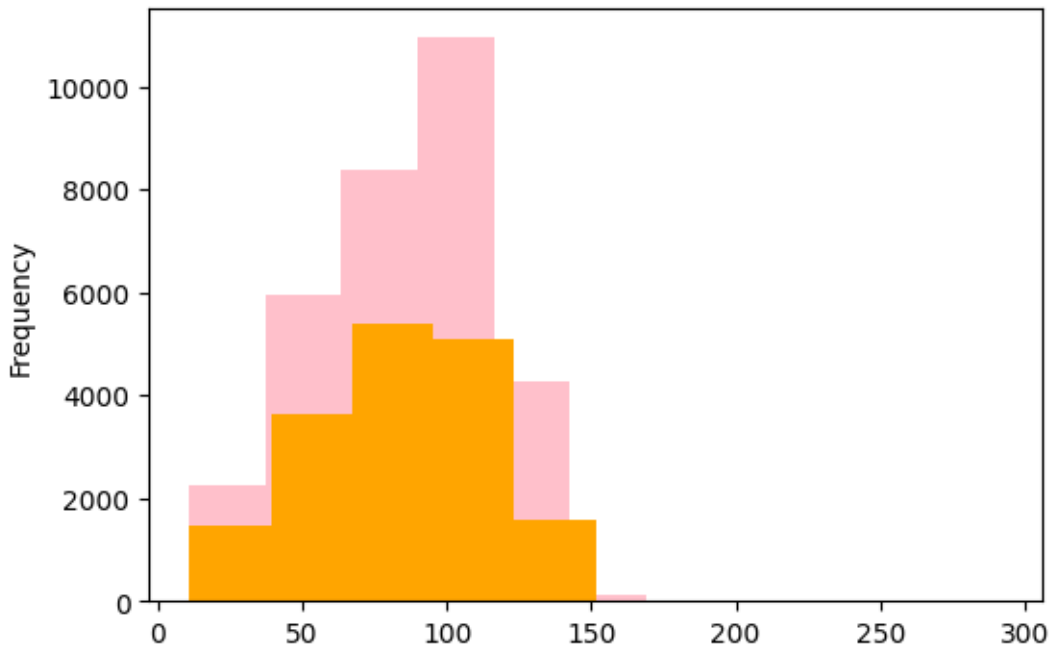
```
23    24      1    @user @user lumpy says i am a . prove it lumpy.
34    35      1  it's unbelievable that in the 21st century we'…
56    57      1            @user lets fight against  #love #peace
68    69      1  ð ©the white establishment can't have blk fol…
77    78      1  @user hey, white people: you can call people '…
82    83      1  how the #altright uses  &amp; insecurity to lu…
111  112      1  @user i'm not interested in a #linguistics tha…
```

[32]:
```
train['label'].value_counts().plot.bar(color = 'pink', figsize = (6, 4))
```

[32]:
```
<Axes: xlabel='label'>
```



[33]:
```
# checking the distribution of tweets in the data

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



[34]:
```
# adding a column to represent the length of the tweet

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

train.head(10)
```

[34]:

	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

[35]:
```
train.groupby('label').describe()
```

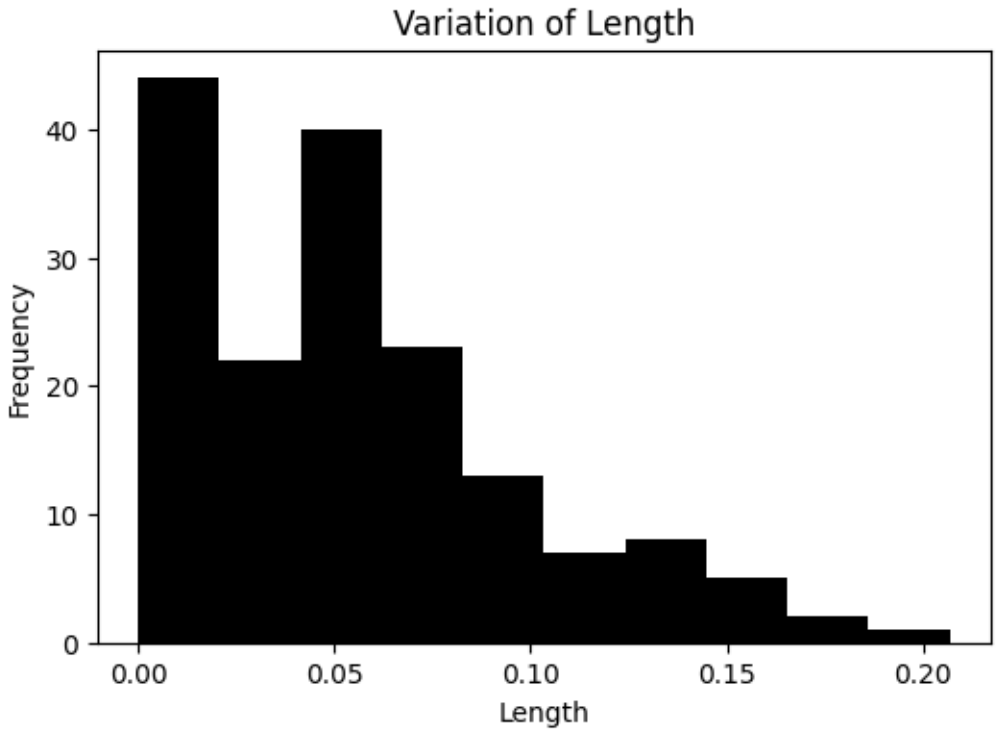
[35]:

	id							\
	count	mean	std	min	25%	50%	75%	
label								
0	29720.0	15974.454441	9223.783469	1.0	7981.75	15971.5	23965.25	

```
1       2242.0  16074.896075  9267.955758  14.0  8075.25  16095.0  24022.00
```

label	len max	len count	len mean	len std	len min	len 25%	len 50%	len 75%	len max
0	31962.0	29720.0	84.328634	29.566484	11.0	62.0	88.0	107.0	274.0
1	31961.0	2242.0	90.187779	27.375502	12.0	69.0	96.0	111.0	152.0

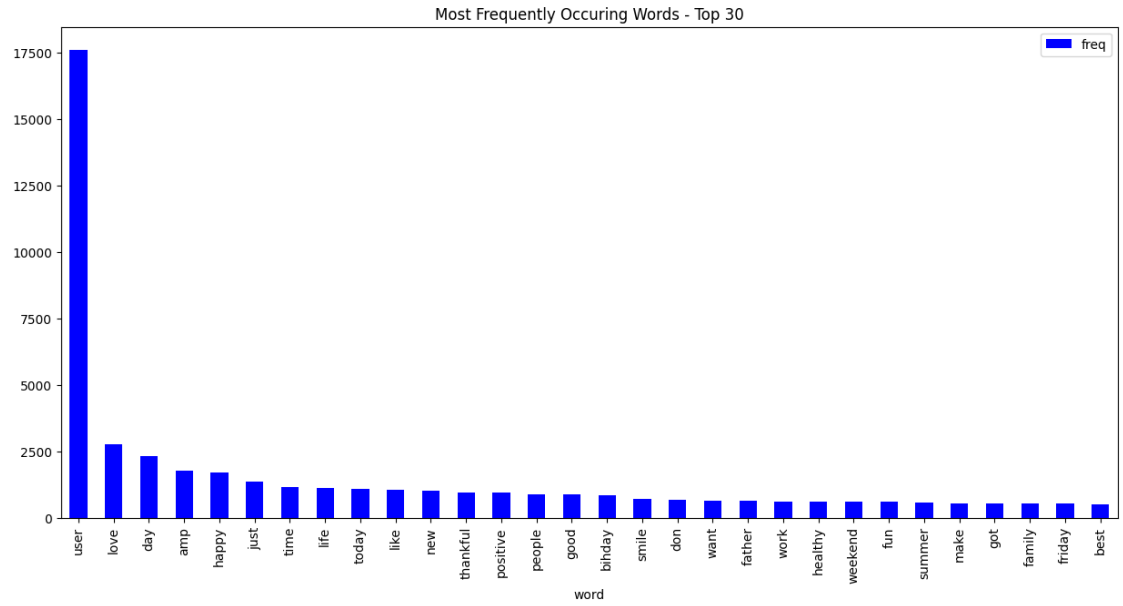
[36]:
```
train['label'] = pd.to_numeric(train['label'], errors='coerce')

# Group by 'len' column and calculate the mean of 'label'
mean_label_by_len = train.groupby('len')['label'].mean()

# Plot histogram
mean_label_by_len.plot.hist(color='black', figsize=(6, 4))
plt.title('Variation of Length')
plt.xlabel('Length')
plt.show()
```



[37]:
```
from sklearn.feature_extraction.text import CountVectorizer


cv = CountVectorizer(stop_words = 'english')
```

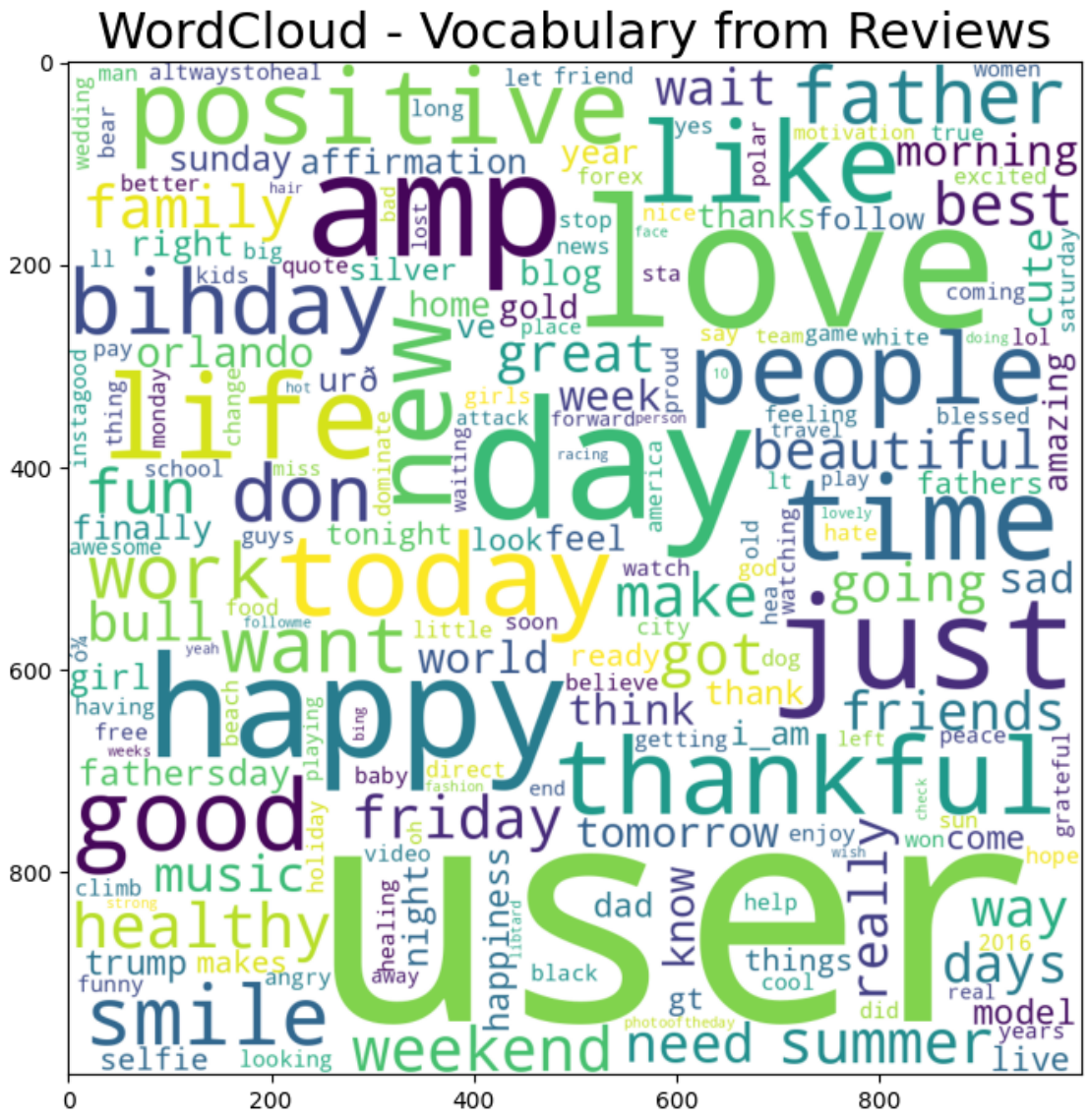
```
words = cv.fit_transform(train.tweet)

sum_words = words.sum(axis=0)

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)

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

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

[37]: Text(0.5, 1.0, 'Most Frequently Occuring Words - Top 30')



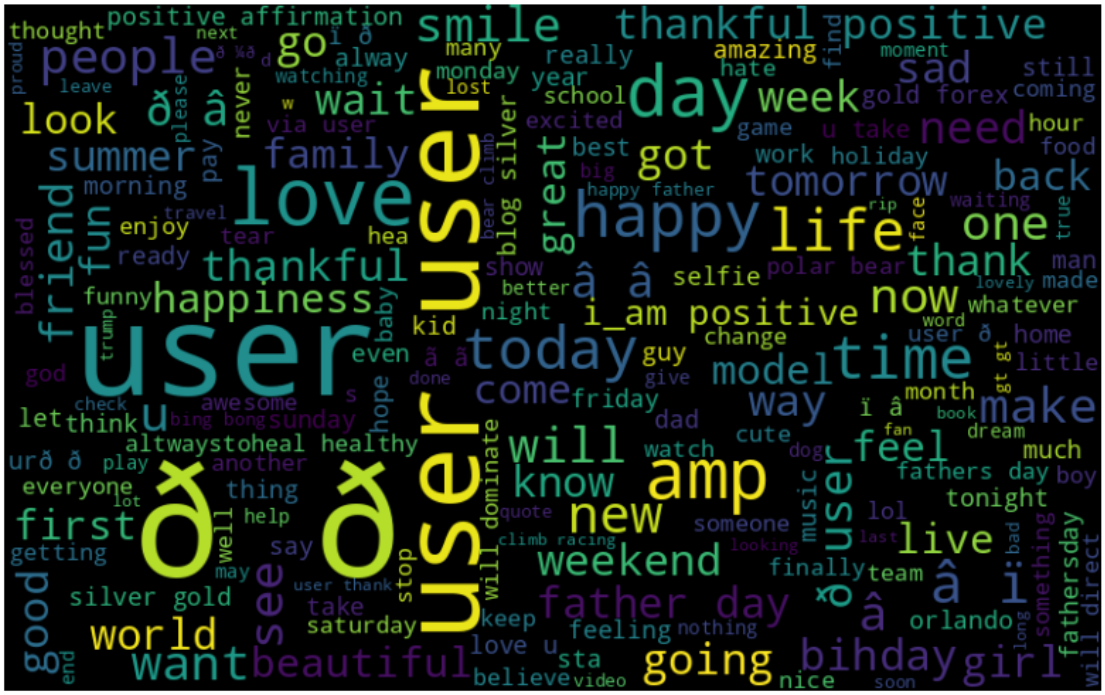
[38]:
```
from wordcloud import WordCloud

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

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

[38]: Text(0.5, 1.0, 'WordCloud - Vocabulary from Reviews')



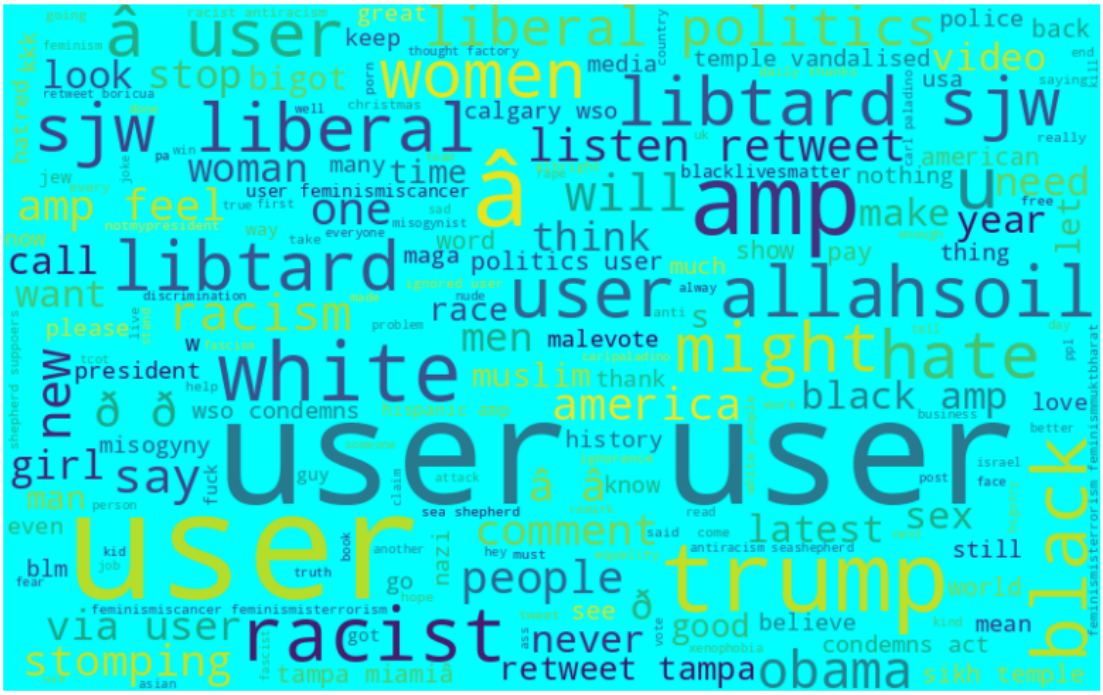
[39]:
```
normal_words =' '.join([text for text in train['tweet'][train['label'] == 0]])

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()
```

The Neutral Words



[40]:
```
negative_words =' '.join([text for text in train['tweet'][train['label'] == 1]])

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()
```

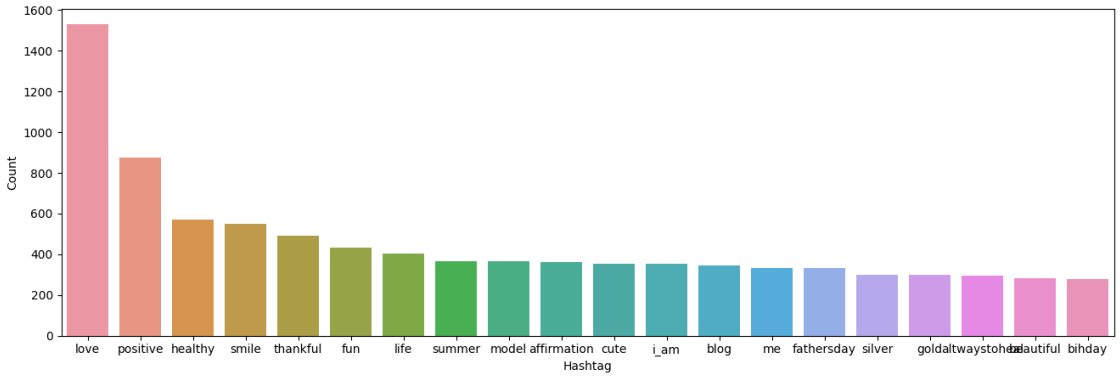
The Negative Words



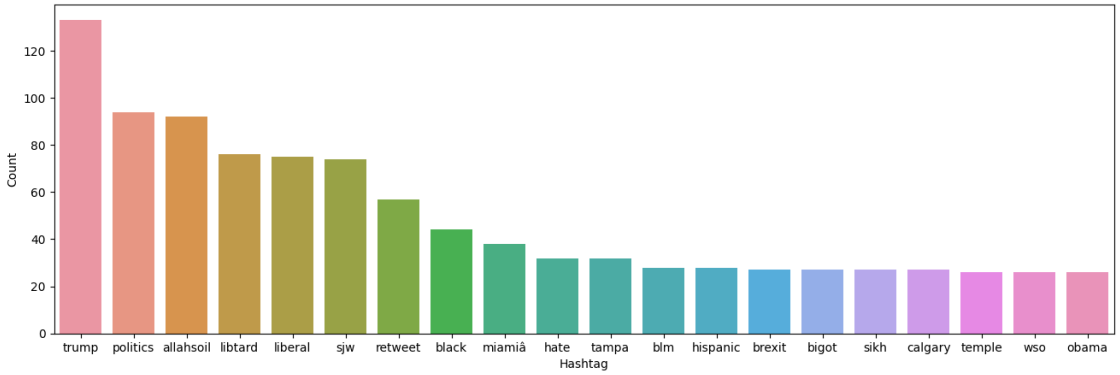
[41]:
```
# collecting the hashtags

def hashtag_extract(x):
    hashtags = []

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

    return hashtags
```

[42]:
```
import re
# extracting hashtags from non racist/sexist tweets
HT_regular = hashtag_extract(train['tweet'][train['label'] == 0])

# extracting hashtags from racist/sexist tweets
HT_negative = hashtag_extract(train['tweet'][train['label'] == 1])

# unnesting list
HT_regular = sum(HT_regular,[])
HT_negative = sum(HT_negative,[])
```

[43]:
```
import nltk
from nltk import FreqDist
a = nltk.FreqDist(HT_regular)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})

# selecting top 20 most frequent hashtags
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()
```



[44]:
```
a = nltk.FreqDist(HT_negative)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})

# selecting top 20 most frequent hashtags
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()
```



[45]:
```
import gensim
from gensim.models import Word2Vec
tokenized_tweet = train['tweet'].apply(lambda x: x.split())
# Assuming tokenized_tweet is your tokenized list of tweets

# Creating a Word2Vec model
model_w2v = Word2Vec(
    sentences=tokenized_tweet,
    vector_size=200,  # Desired number 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,        # Number of cores
    seed=34
)

# Training the Word2Vec model
model_w2v.train(tokenized_tweet, total_examples=len(train['tweet']), epochs=20)
```

[45]:
```
(6109793, 8411580)
```

[46]:
```
model_w2v.wv.most_similar(positive = "dinner")
```

[46]:
```
[('spaghetti', 0.6678438186645508),
 ('#prosecco', 0.6224561333656311),
 ('#wanderlust', 0.6045688986778259),
 ('fluffy', 0.5986981391906738),
 ('#deutschland', 0.5853589177131653),
 ('#restaurant', 0.5850751996040344),
 ('7!', 0.5825372338294983),
 ('#boardgames', 0.5781636238098145),
```

```
 ('#demoday', 0.5778859853744507),
 ('coaching', 0.5774686336517334)]
```

[47]:
```
model_w2v.wv.most_similar(positive = "cancer")
```

[47]:
```
[('champion,', 0.7055912017822266),
 ('level.', 0.6961607336997986),
 ('ways.', 0.6891152262687683),
 ('#merica', 0.6868944764137268),
 ('ownership', 0.6864113211631775),
 ('intelligent', 0.6856280565261841),
 ('tolerance', 0.6843158602714539),
 ('aol', 0.6814162135124207),
 ('spots.', 0.6808521151542664),
 ('law.', 0.6806308627128601)]
```

[48]:
```
model_w2v.wv.most_similar(positive = "apple")
```

[48]:
```
[('mytraining', 0.7102106809616089),
 ('"mytraining"', 0.7083417773246765),
 ('training"', 0.6870191693305969),
 ('app,', 0.6463671922683716),
 ('"my', 0.6045624017715454),
 ('app', 0.592377245426178),
 ('heroku', 0.5815081000328064),
 ('bees', 0.5762739181518555),
 ('ta', 0.5688378810882568),
 ('#expressjs', 0.5680015683174133)]
```

[49]:
```
model_w2v.wv.most_similar(negative = "hate")
```

[49]:
```
[('#staup', 0.017515383660793304),
 ('â\x9c\x88ï¸\x8f', 0.015509642660617828),
 ('#ireland', 0.005592217668890953),
 ('street,', 0.00165042863227427),
 ('#css', 0.00045058902469463646),
 ('#foodie', -0.007494824472814798),
 ('lion', -0.007937184534966946),
 ('#euro2016', -0.008117257617413998),
 ('de', -0.008449590764939785),
 ('#ebay', -0.008534024469554424)]
```

[50]:
```
from tqdm import tqdm
tqdm.pandas(desc="progress-bar")
from gensim.models.doc2vec import TaggedDocument

# Assuming tokenized_tweet is your tokenized list of tweets
```

```
# Tagging documents (sentences or tweets) with unique labels
tagged_data = [TaggedDocument(words=words, tags=[str(i)]) for i, words in␣
 ↪enumerate(tokenized_tweet)]
```

[51]:
```
def add_label(twt):
    output = []
    for i, s in zip(twt.index, twt):
        output.append(TaggedDocument(words=s, tags=["tweet_" + str(i)]))
    return output

# Label all the tweets
labeled_tweets = add_label(tokenized_tweet)

labeled_tweets[:6]
```

[51]:
```
[TaggedDocument(words=['@user', 'when', 'a', 'father', 'is', 'dysfunctional',
'and', 'is', 'so', 'selfish', 'he', 'drags', 'his', 'kids', 'into', 'his',
'dysfunction.', '#run'], tags=['tweet_0']),
 TaggedDocument(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']),
 TaggedDocument(words=['bihday', 'your', 'majesty'], tags=['tweet_2']),
 TaggedDocument(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']),
 TaggedDocument(words=['factsguide:', 'society', 'now', '#motivation'],
tags=['tweet_4']),
 TaggedDocument(words=['[2/2]', 'huge', 'fan', 'fare', 'and', 'big', 'talking',
'before', 'they', 'leave.', 'chaos', 'and', 'pay', 'disputes', 'when', 'they',
'get', 'there.', '#allshowandnogo'], tags=['tweet_5'])]
```

[52]:
```
# removing unwanted patterns from the data

import re
import nltk

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

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[53]:
```
train_corpus = []
```

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

  ps = PorterStemmer()

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

  # joining them back with space
  review = ' '.join(review)
  train_corpus.append(review)
```

[54]:
```
test_corpus = []

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

  ps = PorterStemmer()

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

  # joining them back with space
  review = ' '.join(review)
  test_corpus.append(review)
```

[55]:
```
# creating bag of words

from sklearn.feature_extraction.text import CountVectorizer

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

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

```
(31962, 2500)
(31962,)
```

[56]:
```
# creating bag of words

from sklearn.feature_extraction.text import CountVectorizer

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

print(x_test.shape)
```

```
(17197, 2500)
```

[57]:
```
# splitting the training data into train and valid sets

from sklearn.model_selection import train_test_split

x_train, x_valid, y_train, y_valid = train_test_split(x, y, test_size = 0.25,␣
 ↪random_state = 42)

print(x_train.shape)
print(x_valid.shape)
print(y_train.shape)
print(y_valid.shape)
```

```
(23971, 2500)
(7991, 2500)
(23971,)
(7991,)
```

[58]:
```
# standardization

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train = sc.fit_transform(x_train)
x_valid = sc.transform(x_valid)
x_test = sc.transform(x_test)
```

[59]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score

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)
```

```
Training Accuracy : 0.9991656585040257
Validation Accuracy : 0.9525716430984857
F1 score : 0.6160081053698074
[[7308  124]
 [ 255  304]]
```

[60]:
```
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 : 0.9851487213716574
Validation Accuracy : 0.9416843949443123
f1 score : 0.5933682373472949
[[7185  247]
 [ 219  340]]
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
```

```
  n_iter_i = _check_optimize_result(
```

[61]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score, confusion_matrix

# Assuming x_train, x_valid, y_train, and y_valid are defined and contain your␣
 ↪training and validation data

# Instantiate Decision Tree classifier
model = DecisionTreeClassifier()

# Train the model
model.fit(x_train, y_train)

# Predictions on the validation set
y_pred = model.predict(x_valid)

# Print training and validation accuracy
print("Training Accuracy:", model.score(x_train, y_train))
print("Validation Accuracy:", model.score(x_valid, y_valid))

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

# Generate confusion matrix
cm = confusion_matrix(y_valid, y_pred)
print("Confusion Matrix:")
print(cm)
```

```
Training Accuracy: 0.9991656585040257
Validation Accuracy: 0.9343010887248154
F1 score: 0.5485812553740327
Confusion Matrix:
[[7147  285]
 [ 240  319]]
```

[62]:
```
from sklearn.svm import SVC

model = SVC()
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 : 0.978181969880272
Validation Accuracy : 0.9521962207483419
f1 score : 0.4986876640419947
[[7419   13]
 [ 369  190]]
```

[63]:
```
from xgboost import XGBClassifier

model = XGBClassifier()
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)t
print(cm)
```

```
Training Accuracy : 0.9608693838388053
Validation Accuracy : 0.9550744587661119
f1 score : 0.575147928994083
[[7389   43]
 [ 316  243]]
```