Whatsapp Status Sentiment Analysis Analysis

In [1]:
```
#Importing pandas and numpy library
import numpy as np
import pandas as pd
```

In [11]:
```
#Loading the sad happy and angry datasets
sad = pd.read_csv('sad.csv')
happy= pd.read_csv('happy.csv')
angry=pd.read_csv('angry.csv')
```

In [4]:
```
#Finding the shapes of all datas
print("Happy status data: ",happy.shape)
print("Angry status data: ",angry.shape)
print("Sad status data: ",sad.shape)
```

```
Happy status data:  (708, 2)
Angry status data:  (696, 2)
Sad status data:  (635, 2)
```

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

Out[5]:

	content	sentiment
0	Wants to know how the hell I can remember word...	happy
1	Love is a long sweet dream & marriage is an al...	happy
2	The world could be amazing when you are slight...	happy
3	My secret talent is getting tired without doin...	happy
4	Khatarnaak Whatsapp Status Ever… Can\'t talk, ...	happy

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

Out[6]:

	content	sentiment
0	Sometimes I'm not angry, I'm hurt and there's ...	angry
1	Not available for busy people☺	angry
2	I do not exist to impress the world. I exist t...	angry
3	Everything is getting expensive except some pe...	angry
4	My phone screen is brighter than my future ☹	angry

In [7]:
```
sad.head()
```

Out[7]:

	content	sentiment
0	Never hurt people who love you a lot, because ...	sad
1	Don’t expect me to tell you what you did wrong...	sad
2	I preferred walking away than fighting for you...	sad
3	Moving forward in life isn’t the hard part, it...	sad
4	Never cry for anyone in your life, because tho...	sad

In [12]:
```
#Dropping the duplicates from all the datas
happy.drop_duplicates(subset='content', keep="first",inplace=True)
angry.drop_duplicates(subset='content', keep="first",inplace=True)
sad.drop_duplicates(subset='content', keep="first",inplace=True)
```

In [13]:
```
#Finding the shapes of all datas after removing duplicates
print("Happy status data: ",happy.shape)
print("Angry status data: ",angry.shape)
print("Sad status data: ",sad.shape)
```

```
Happy status data:  (704, 2)
Angry status data:  (498, 2)
Sad status data:  (390, 2)
```

In [14]:
```
#Concatenating the three datas in to df
df = pd.concat([angry, happy, sad])
```

In [16]:
```
#Displaying the concatenated dataframe
df.sample(5)
```

Out[16]:

	content	sentiment
17	To the less fortunate… life is nothing but a t...	sad
302	You are only as miserable as you perceive to b...	happy
244	Its never easy missing someone and knowing you...	sad
495	I don’t regret my past. I just regret the time...	angry
310	It hurts when you realize you aren't as import...	sad

In [17]:
```
#Looking for the info of df
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1592 entries, 0 to 634
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   content    1592 non-null   object
 1   sentiment  1592 non-null   object
dtypes: object(2)
memory usage: 37.3+ KB
```

In [22]:
```
#check for na's
df['content'].replace('', np.nan, inplace=True)
df.isna().sum()
```

Out[22]:
```
content      0
sentiment    0
dtype: int64
```

In [18]:
```
#Checking for count on each unique item
df['sentiment'].value_counts()
```

Out[18]:
```
happy    704
angry    498
sad      390
Name: sentiment, dtype: int64
```

Preprocessing Text

In [27]:
```
import re
import string

contractions = {
"ain't": "am not",
"aren't": "are not",
"can't": "cannot",
"can't've": "cannot have",
"'cause": "because",
"could've": "could have",
"couldn't": "could not",
"couldn't've": "could not have",
"didn't": "did not",
"doesn't": "does not",
"don't": "do not",
"hadn't": "had not",
"hadn't've": "had not have",
"hasn't": "has not",
"haven't": "have not",
"he'd": "he would",
"he'd've": "he would have",
"he'll": "he will",
"he's": "he is",
"how'd": "how did",
"how'll": "how will",
"how's": "how is",
"i'd": "i would",
"i'll": "i will",
"i'm": "i am",
"i've": "i have",
"isn't": "is not",
"it'd": "it would",
"it'll": "it will",
"it's": "it is",
"let's": "let us",
"ma'am": "madam",
"mayn't": "may not",
"might've": "might have",
"mightn't": "might not",
"must've": "must have",
"mustn't": "must not",
"needn't": "need not",
"oughtn't": "ought not",
"shan't": "shall not",
"sha'n't": "shall not",
"she'd": "she would",
"she'll": "she will",
"she's": "she is",
"should've": "should have",
"shouldn't": "should not",
"that'd": "that would",
"that's": "that is",
"there'd": "there had",
"there's": "there is",
"they'd": "they would",
"they'll": "they will",
"they're": "they are",
"they've": "they have",
"wasn't": "was not",
"we'd": "we would",
"we'll": "we will",
"we're": "we are",
"we've": "we have",
"weren't": "were not",
"what'll": "what will",
"what're": "what are",
"what's": "what is",
"what've": "what have",
"where'd": "where did",
"where's": "where is",
"who'll": "who will",
"who's": "who is",
"won't": "will not",
"wouldn't": "would not",
"you'd": "you would",
"you'll": "you will",
"you're": "you are",
"thx"   : "thanks"
}


def clean(text):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", flags=re.UNICODE)
    text = emoji_pattern.sub(r'', text)
    text = text.lower()
    text = re.sub('\[.*?\]','',text)  # remove tings in brackets []
    text = re.sub('https?://\S+|www\.\S+','',text) #remove website links
    text = re.sub('<,*?>+"','',text)   #remove html tags
   ### text = re.sub('[%s]' % re.escape(string.punctuation),'',text)
    text = re.sub('\n','',text) # remove line charchters
    text = re.sub(u"\U00002019", "'", text) # IMPORTANT: Their apostrophe charact
    words = text.split()
    for i in range(len(words)):
        if words[i].lower() in contractions.keys():
            words[i] = contractions[words[i].lower()]
    text = " ".join(words)
    return text

df['content'] = df['content'].apply(lambda x: clean(x))
```

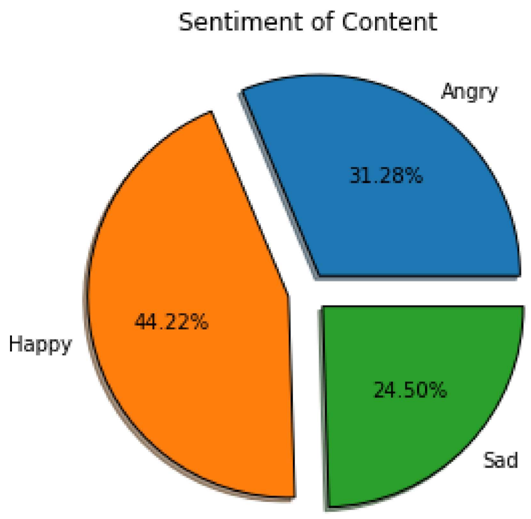
EDA

In [23]:
```
from matplotlib import pyplot as plt
import seaborn as sns
```

In [25]:
```
total = df.shape[0]
values = df['sentiment'].value_counts()

num_angry = values['angry']
num_happy = values['happy']
num_sad = values['sad']

slices = [num_angry, num_happy, num_sad]
labeling = ['Angry','Happy', 'Sad']
explode = [0.1, 0.1, 0.1]
plt.pie(slices,explode=explode,shadow=True,autopct='%1.2f%%',labels=labeling,wedg
plt.title('Sentiment of Content')
plt.tight_layout()
plt.show()
```



In [ ]:

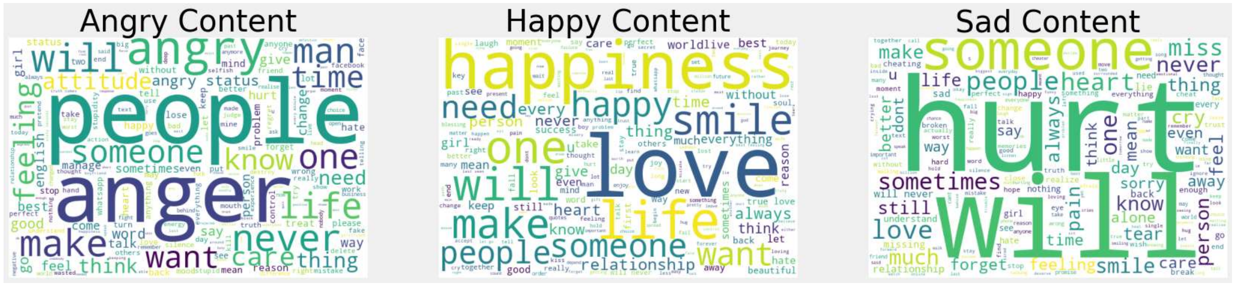
In [33]:
```
from wordcloud import WordCloud,STOPWORDS

stop_word= list(STOPWORDS)

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=[26, 10])
wordcloud1 = WordCloud( background_color='white',stopwords = stop_word,
                        width=600,
                        height=400).generate(" ".join(df[df['sentiment']=='angry'
ax1.imshow(wordcloud1)
ax1.axis('off')
ax1.set_title('Angry Content',fontsize=40)

wordcloud2 = WordCloud( background_color='white',stopwords = stop_word,
                        width=600,
                        height=400).generate(" ".join(df[df['sentiment']=='happy'
ax2.imshow(wordcloud2)
ax2.axis('off')
ax2.set_title('Happy Content',fontsize=40)

wordcloud3 = WordCloud( background_color='white',stopwords = stop_word,
                        width=600,
                        height=400).generate(" ".join(df[df['sentiment']=='sad']|
ax3.imshow(wordcloud3)
ax3.axis('off')
ax3.set_title('Sad Content',fontsize=40)
plt.show()
```



In [ ]:

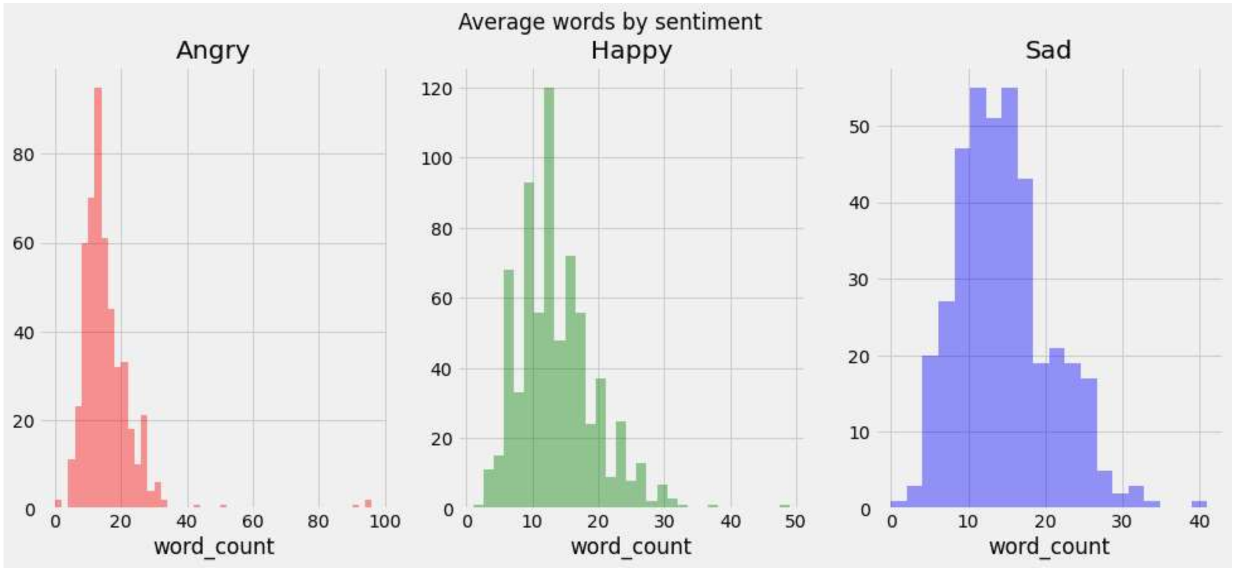
In [43]:
```
#word count
df['word_count'] = df['content'].apply(lambda x: len(x.split()))


fig,(ax1,ax2,ax3) = plt.subplots(1,3,figsize=(15,6))
df_angry = df[df['sentiment'] == 'angry']
word = df_angry['word_count']
sns.distplot(word,ax=ax1,color='red', kde=False)
ax1.set_title('Angry')

df_happy = df[df['sentiment'] == 'happy']
word = df_happy['word_count']
sns.distplot(word,ax=ax2,color='green', kde=False)
ax2.set_title('Happy')

df_sad = df[df['sentiment'] == 'sad']
word = df_sad['word_count']
sns.distplot(word,ax=ax3,color='blue', kde=False)
ax3.set_title('Sad')

fig.suptitle('Average words by sentiment')
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

```

In [45]:

```
#replacing the categorical values
df['sentiment'].replace({'angry':0,'happy':1,'sad':2},inplace=True)
```

In [46]:

```
#Categorical converted to numerial
df['sentiment'].value_counts()
```

Out[46]:

```
1    704
0    498
2    390
Name: sentiment, dtype: int64
```

Removing stop words

In [48]:

```
import nltk
#nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))


def remove_stopwords(text):
    text = text.split()
    words = [w for w in text if w not in stopwords.words('english')]
    return " ".join(words)

df['content_no_stopwords'] = df['content'].apply(lambda x : remove_stopwords(x))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ABC\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

In [49]:

```
df.head()
```

Out[49]:

	content	sentiment	word_count	content_no_sw
0	sometimes i am not angry, i am hurt and there ...	0	14	sometimes angry, hurt big difference.
1	not available for busy people	0	5	available busy people
2	i do not exist to impress the world. i exist t...	0	22	exist impress world. exist live life way make ...
3	everything is getting expensive except some pe...	0	11	everything getting expensive except people, ge...
4	my phone screen is brighter than my future	0	8	phone screen brighter future

In [ ]:

```

```

Lemmatizing and Stemming

In [55]:
```
# Lemmatizing and Stemming
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import sent_tokenize,word_tokenize

lemmatizer = WordNetLemmatizer()
statuses = df['content'].values.copy()

for i in range(len(statuses)):
    a = statuses[i]
    sentences = sent_tokenize(statuses[i])
    word_list = []
    for sent in sentences:
        words = word_tokenize(sent)
        for word in words:
            if words not in word_list:
                word_list.append(word)
    word_list = [lemmatizer.lemmatize(w) for w in word_list if w not in stop_wor
    statuses[i] = ' '.join(w for w in word_list)

from nltk.stem import PorterStemmer
porter = PorterStemmer()

for i in range(len(statuses)):
    sentences = sent_tokenize(statuses[i])
    word_list = []
    for sent in sentences:
        words = word_tokenize(sent)
        for word in words:
            if words not in word_list:
                word_list.append(word)
    word_list = [porter.stem(w) for w in word_list if w not in stop_words]
    statuses[i] = ' '.join(w for w in word_list)


df['content_stem_lem'] = statuses
```

In [56]:
```
df.head()
```

Out[56]:

	content	sentiment	word_count	content_no_sw	content_stem_lem
0	sometimes i am not angry, i am hurt and there ...	0	14	sometimes angry, hurt big difference.	sometim angri , hurt big differ .
1	not available for busy people	0	5	available busy people	avail busi peopl
2	i do not exist to impress the world. i exist t...	0	22	exist impress world. exist live life way make ...	exist impress world . exist live life way make...
3	everything is getting expensive except some pe...	0	11	everything getting expensive except people, ge...	everyth get expens except peopl , get cheaper .
4	my phone screen is brighter than my future	0	8	phone screen brighter future	phone screen brighter futur

Test train split

In [62]:
```
X = df.iloc[:,-1].values
y = df.sentiment.values
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test = train_test_split(X,y,test_size=0.2)
```

Feature Extraction

In [64]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=3000)
X_train_cv = cv.fit_transform(X_train).toarray()
X_test_cv = cv.transform(X_test).toarray()
```

In [67]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(min_df= 1, max_features=1500, strip_accents='unicode',ana
                     ngram_range=(1,3),stop_words='english')
X_train_tfidf = tfidf.fit_transform(X_train).toarray()
X_test_tfidf = tfidf.transform(X_test).toarray()
```

In [ ]:

Classification

In [68]:
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
clf1=GaussianNB()
clf2=MultinomialNB()
clf3=BernoulliNB()
```

In [71]:
```
clf1.fit(X_train_cv,y_train)
clf2.fit(X_train_cv,y_train)
clf3.fit(X_train_cv,y_train)
y_pred1=clf1.predict(X_test_cv)
y_pred2=clf2.predict(X_test_cv)
y_pred3=clf3.predict(X_test_cv)
```

In [74]:
```
from sklearn.metrics import accuracy_score
print("Gaussian Cv",accuracy_score(y_test,y_pred1))
print("Multinomial CV",accuracy_score(y_test,y_pred2))
print("Bernaulli CV",accuracy_score(y_test,y_pred3))
```

```
Gaussian Cv 0.5391849529780565
Multinomial CV 0.7053291536050157
Bernaulli CV 0.6865203761755486
```

In [77]:
```
clf4=GaussianNB()
clf5=MultinomialNB()
clf6=BernoulliNB()
clf4.fit(X_train_tfidf,y_train)
clf5.fit(X_train_tfidf,y_train)
clf6.fit(X_train_tfidf,y_train)
y_pred4=clf4.predict(X_test_tfidf)
y_pred5=clf5.predict(X_test_tfidf)
y_pred6=clf6.predict(X_test_tfidf)
```

In [78]:
```
print("Gaussian TFIDF",accuracy_score(y_test,y_pred4))
print("Multinomial TFIDF",accuracy_score(y_test,y_pred5))
print("Bernaulli TFIDF",accuracy_score(y_test,y_pred6))
```

```
Gaussian TFIDF 0.5297805642633229
Multinomial TFIDF 0.6833855799373041
Bernaulli TFIDF 0.6238244514106583
```

In [79]:
```
#testing the data for a sentence
test="i am sad & depressed too"
```

In [80]:
```
test = np.array([test])
test = cv.transform(test)
```

In [81]:
```
#Here 2 implicates the Sad
clf2.predict(test)
```

Out[81]:
```
array([2], dtype=int64)
```

In [ ]:
```

```

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

In [86]:
```

clf_rf1=RandomForestClassifier(n_estimators=500)
clf_rf1.fit(X_train_cv,y_train)
y_pred_rf1=clf_rf1.predict(X_test_cv)
print("Random Forest CV",accuracy_score(y_test,y_pred_rf1))
```

```
Random Forest CV 0.7272727272727273
```

In [88]:
```
from sklearn.ensemble import RandomForestClassifier
clf_rf2=RandomForestClassifier(n_estimators=500)
clf_rf2.fit(X_train_tfidf,y_train)
y_pred_rf2=clf_rf2.predict(X_test_tfidf)
print("Random Forest TFIDF",accuracy_score(y_test,y_pred_rf2))
```

```
Random Forest TFIDF 0.7115987460815048
```

```
In [ ]:
```