Twitter US Airline Sentiment

Background and Context:

Twitter possesses 330 million monthly active users, which allows businesses to reach a broad population and connect with customers without intermediaries. On the other hand, there’s so much information that it’s difficult for brands to quickly detect negative social mentions that could harm their business.

That's why sentiment analysis/classification, which involves monitoring emotions in conversations on social media platforms, has become a key strategy in social media marketing.

Listening to how customers feel about the product/service on Twitter allows companies to understand their audience, keep on top of what’s being said about their brand and their competitors, and discover new trends in the industry.

Data Description:

A sentiment analysis job about the problems of each major U.S. airline. Twitter data was scraped from February of 2015 and contributors were asked to first classify positive, negative, and neutral tweets, followed by categorizing negative reasons (such as "late flight" or "rude service").

Dataset:

The dataset has the following columns:

- tweet_id
- airline_sentiment
- airline_sentiment_confidence
- negativereason
- negativereason_confidence
- airline
- airline_sentiment_gold
- name
- negativereason_gold
- retweet_count
- text
- tweet_coord
- tweet_created
- tweet_location
- user_timezone

Data Summary

Install and import necessary libraries.

In [ ]:
```
!pip install contractions

import re, string, unicodedata                          # Import Rege
x, string and unicodedata.
import contractions                                     # Import contr
actions library.
from bs4 import BeautifulSoup                           # Import Beaut
ifulSoup.

import seaborn as sns
import numpy as np                                      # Import nump
y.
import pandas as pd                                     # Import panda
s.
import nltk                                             # Import Natur
al Language Tool-Kit.
from google.colab import drive
from tqdm import tqdm

nltk.download('stopwords')                              # Download Sto
pwords.
nltk.download('punkt')
nltk.download('wordnet')


from nltk.corpus import stopwords                       # Import stopw
ords.
from nltk.tokenize import word_tokenize, sent_tokenize  # Import Token
izer.
from nltk.stem.wordnet import WordNetLemmatizer         # Import Lemma
tizer.
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.de
v/colab-wheels/public/simple/
Requirement already satisfied: contractions in /usr/local/lib/python3.
7/dist-packages (0.1.72)
Requirement already satisfied: textsearch>=0.0.21 in /usr/local/lib/py
thon3.7/dist-packages (from contractions) (0.0.21)
Requirement already satisfied: anyascii in /usr/local/lib/python3.7/di
st-packages (from textsearch>=0.0.21->contractions) (0.3.1)
Requirement already satisfied: pyahocorasick in /usr/local/lib/python
3.7/dist-packages (from textsearch>=0.0.21->contractions) (1.4.4)
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [ ]:

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [162]:

```
data=pd.read_csv('/content/drive/MyDrive/MachineLearning/TwitterUSAirl
ineSentimentData/Tweets.csv')
```

In [ ]:

```
data.shape                                                # print shape
of data.
```

Out[ ]:

```
(14640, 15)
```

Observations:

- Our csv file has 14640 records and 15 columns

In [ ]:
```
data.head()                                                        # Print first
5 rows of data.
```

Out[ ]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativer
0	570306133677760513	neutral	1.0000	NaN	
1	570301130888122368	positive	0.3486	NaN	
2	570301083672813571	neutral	0.6837	NaN	
3	570301031407624196	negative	1.0000	Bad Flight	
4	570300817074462722	negative	1.0000	Can't Tell	

Removing duplicates

In [136]:
```
duplicateRows = data[data.duplicated(keep=False)]
duplicateRows.sort_values("tweet_id", inplace = True)
duplicateRows.shape
duplicateRows.head(6)
```

Out[136]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	nega
12001	570272018840428544	neutral	1.0	NaN	
12162	570272018840428544	neutral	1.0	NaN	
12159	570272880556011520	positive	1.0	NaN	
11998	570272880556011520	positive	1.0	NaN	
11997	570273710210469888	positive	1.0	NaN	
12158	570273710210469888	positive	1.0	NaN	

Observations:

- As we can see, the above table contains duplicate records which needs to be handled.

In [137]:
```
data.drop_duplicates(keep='first',inplace=True)
```

In [ ]:
```
data.shape
```

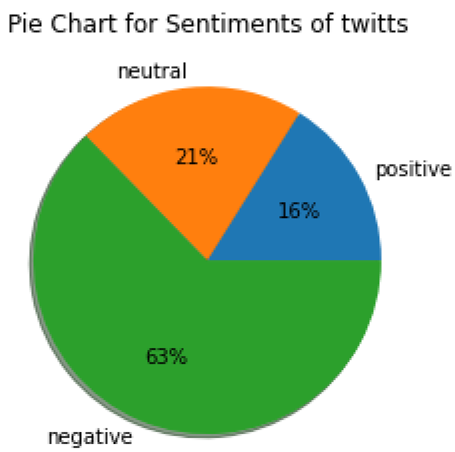
Out[ ]: (14604, 15)

Exploratory data analysis

Plotting

Pie chart for Airline Sentiments

In [ ]:
```
pie_chart = new_data.groupby('airline_sentiment').agg('count')
plt.pie(pie_chart.text.sort_values(), labels=pie_chart.text.sort_values().index,autopct='%.0f%%', shadow=True )
plt.title("Pie Chart for Sentiments of twitts")
plt.show()
```



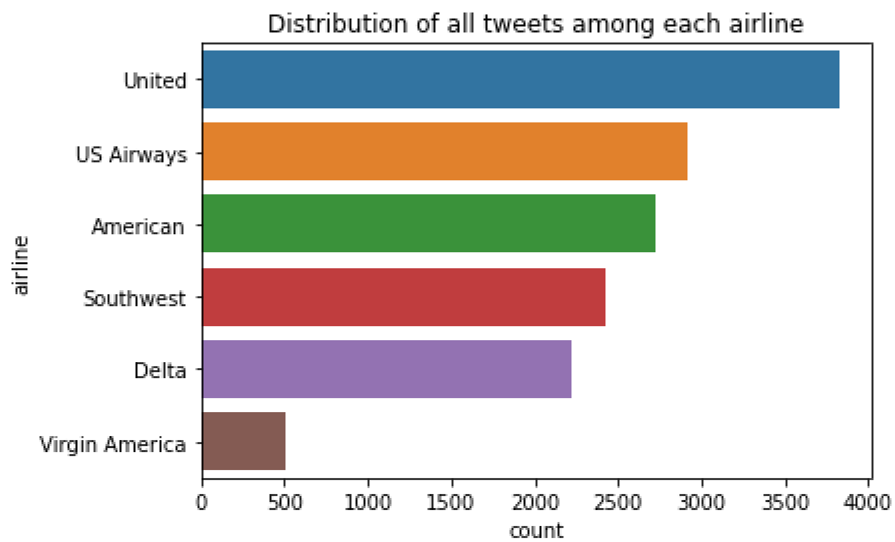
Observations:

- 63% of negative sentiments found in the corpus. Its higher than positive and neutral sentiments.

Distribution of all tweets among each airline

In [ ]:
```
#number of tweets
sns.countplot(data=data,y=data['airline'],order = data['airline'].value_counts().index).set_title('Distribution of all tweets among each airline')
```

Out[ ]: `Text(0.5, 1.0, 'Distribution of all tweets among each airline')`



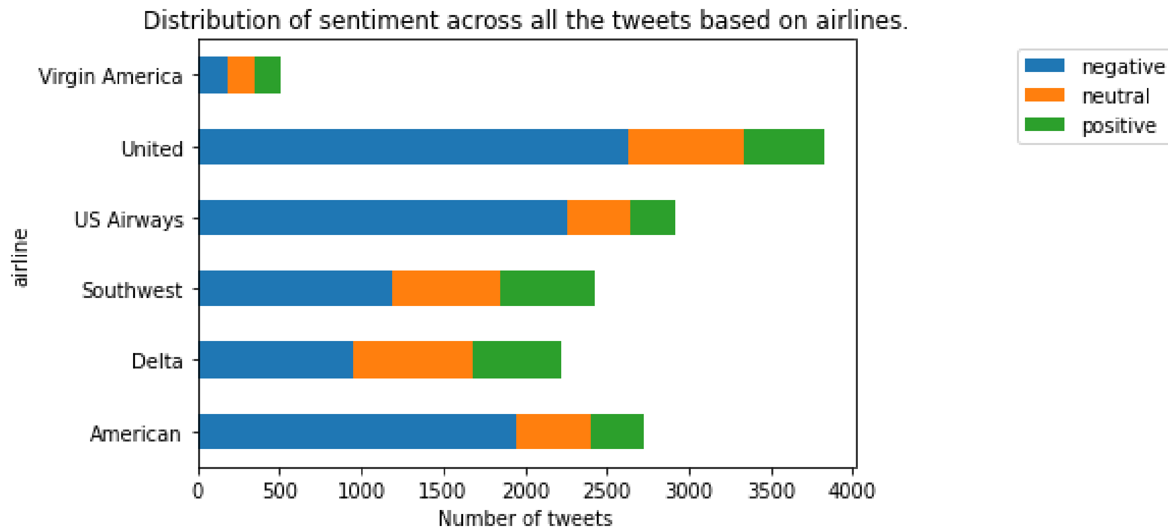
Observations:

- United Airline has maximum number of tweets.
- Virgin America Airline has minimum number of tweets.

Distribution of sentiment across all the tweets.

In [ ]:
```
types = data.groupby("airline")['airline_sentiment'].value_counts(norm
alize=False).sort_index()
types.unstack().plot(kind='barh', stacked='True')
plt.legend(bbox_to_anchor=(1.5, 1), loc='upper right')
plt.xlabel("Number of tweets")
plt.title("Distribution of sentiment across all the tweets based on ai
rlines.")
```

Out[ ]:
```
Text(0.5, 1.0, 'Distribution of sentiment across all the tweets based
on airlines.')
```



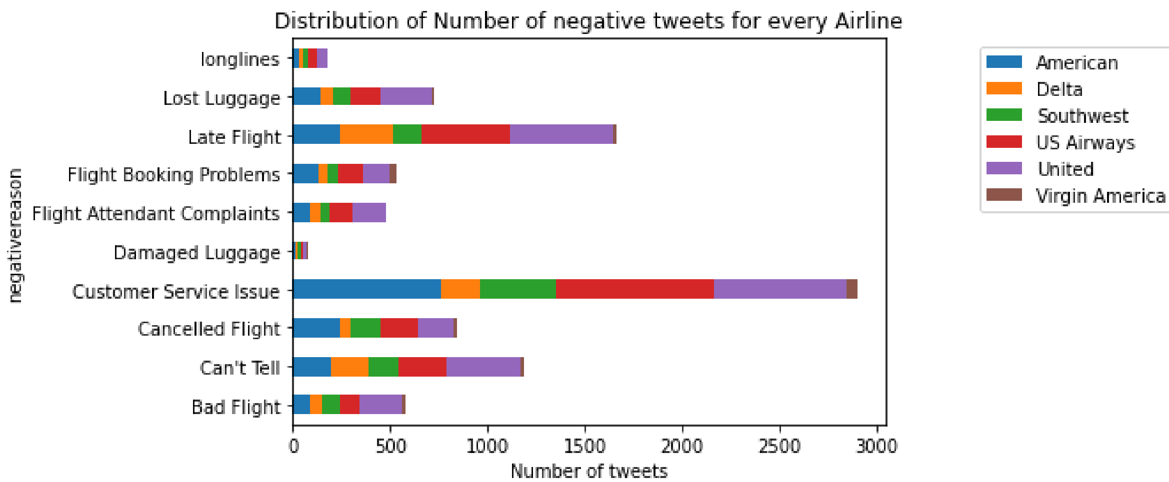
Observations:

- United Airline has more number of negative tweets than neutral and positive combined.It is same for US Airways and American airways.
- Virgin America Airline which has lowest number of tweets has almost similar number of negative, neutral and positive tweets.

Distribution of Number of negative tweets for every Airline

In [ ]:
```
types = data.groupby("negativereason")['airline'].value_counts(normalize=False).sort_index()
types.unstack().plot(kind='barh', stacked='True')
plt.legend(bbox_to_anchor=(1.5, 1), loc='upper right')
plt.xlabel('Number of tweets')
plt.title('Distribution of Number of negative tweets for every Airline')
```

Out[ ]:
```
Text(0.5, 1.0, 'Distribution of Number of negative tweets for every Airline')
```



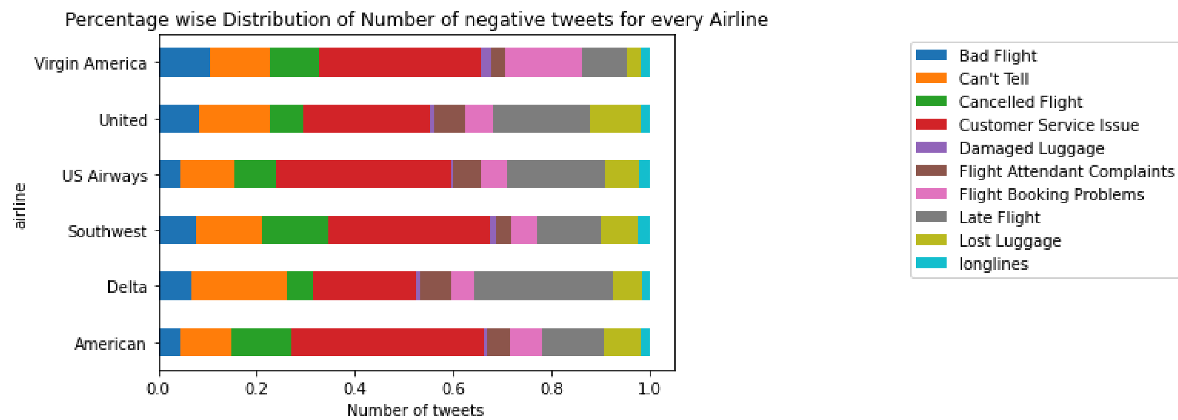
Observations:

- More number of people have suffered from Customer Serive than any other reason
- Least number of people have suffered from late flight

Distribution of all the negative reasons.

In [ ]:
```
types = data.groupby("airline")['negativereason'].value_counts(normalize=True).unstack()
types.plot(kind='barh', stacked='True')
plt.legend(bbox_to_anchor=(2, 1), loc='upper right')
plt.xlabel('Number of tweets')
plt.title('Distribution of all the negative reasons.')
```

Out[ ]:
```
Text(0.5, 1.0, 'Percentage wise Distribution of Number of negative tweets for every Airline')
```



Observations:

- Delta have better customer service than anyother airlines which is most common issues across all the airlines but they have higher late flight issue than any other airlines.
- Virgin America need to work on the customer service and flight booking problems.
- United airlines, US airways, Southwest and American airlines have to work on the customer service and late flight issue issue.

Understanding of Data Columns

In [164]:
```
# Only keeping relevant columns from the data, as these are useful for our analysis as per the requirments.
new_data=data[['airline_sentiment','text']]
```

In [131]:
```
new_data.shape
```

Out[131]:
```
(14604, 2)
```

In [ ]:
```
new_data.head()
```

Out[ ]:

	airline_sentiment	text
0	neutral	@VirginAmerica What @dhepburn said.
1	positive	@VirginAmerica plus you've added commercials t...
2	neutral	@VirginAmerica I didn't today... Must mean I n...
3	negative	@VirginAmerica it's really aggressive to blast...
4	negative	@VirginAmerica and it's a really big bad thing...

In [ ]:
```
new_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14604 entries, 0 to 14639
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   airline_sentiment  14604 non-null  object
 1   text               14604 non-null  object
dtypes: object(2)
memory usage: 858.3+ KB
```

Data Pre - Processing

Decontractions

In [139]:
```
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

Removal of stopwords

In [140]:

```
# Removing the words from the stop words list: 'no', 'nor', 'not'
# Adding "@VirginAmerica", "@united", "@SouthwestAir", "@JetBlue", '@AmericanAir', '@USAirways'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't", "virginamerica", "united", "southwestair", "jetblue", 'americanair', 'usairways' ]
```

Removal of Special Characters and Punctuations

Conversion to lowercase

In [165]:
```
def preprocess_text(text_data):
    preprocessed_text = []
    # tqdm is for printing the status bar
    for sentance in tqdm(text_data):
        sent = decontracted(sentance)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\n', ' ')
        sent = sent.replace('\\"', ' ')
        sent = sent.replace('http"', ' ')          # added later
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
        preprocessed_text.append(sent.lower().strip())
    return preprocessed_text
```

In [166]:
```
# Create a new column for clean text
new_data['clean_text']=preprocess_text(new_data['text'].values)
```

```
100%|██████████| 14640/14640 [00:01<00:00, 13834.64it/s]
```

In [167]:
```
new_data.head()
```

Out[167]:

	airline_sentiment	text	clean_text
0	neutral	@VirginAmerica What @dhepburn said.	dhepburn said
1	positive	@VirginAmerica plus you've added commercials t...	plus added commercials experience tacky
2	neutral	@VirginAmerica I didn't today... Must mean I n...	not today must mean need take another trip
3	negative	@VirginAmerica it's really aggressive to blast...	really aggressive blast obnoxious entertainmen...
4	negative	@VirginAmerica and it's a really big bad thing...	really big bad thing

In [168]:
```
# drop the original column of text
new_data=new_data.drop(['text'],axis=1)
```

In [169]:
```
new_data[new_data['clean_text'].isna()]
```

Out[169]:

airline_sentiment	clean_text

In [170]:
```
# Considaring neutral sentiments as positive sentiments
new_data['airline_sentiment']=new_data['airline_sentiment'].str.replac
e('neutral','positive')
```

In [171]:
```
# convert class label into numerical number
# 1 is used for negative tweets so that it will reflect in recall scor
e
new_data['airline_sentiment'].replace(to_replace='positive', value=0,
inplace=True)
new_data['airline_sentiment'].replace(to_replace='negative', value=1,
inplace=True)
new_data.head()
```

Out[171]:

	airline_sentiment	clean_text
0	0	dhepburn said
1	0	plus added commercials experience tacky
2	0	not today must mean need take another trip
3	1	really aggressive blast obnoxious entertainmen...
4	1	really big bad thing

Analysis of frequency of words

In [172]:
```
txt = ' '.join(new_data['clean_text'])
txt=txt.split()
freq_cnt = pd.Series(txt).value_counts()
type(freq_cnt)
```

Out[172]: pandas.core.series.Series

In [173]:
```
freq_wds = freq_cnt.to_frame()
```

In [174]:
```
freq_wds.tail()
```

Out[174]:

	0
898	1
peanutsonaplatter	1
380	1
1708	1
blackberry10	1

In [175]:
```
freq=new_data['clean_text'].str.split(expand=True).stack().value_count
s().to_frame()
freq.rename(columns = {0:'count'}, inplace = True)

freq.head()
```

Out[175]:

	count
flight	3939
not	3658
no	1508
get	1340
co	1214

Observations:

- flight and not are the two words that are used for more than 3000 times

In [176]:
```
print("Total number of words in Corpus are ",freq['count'].sum())
```

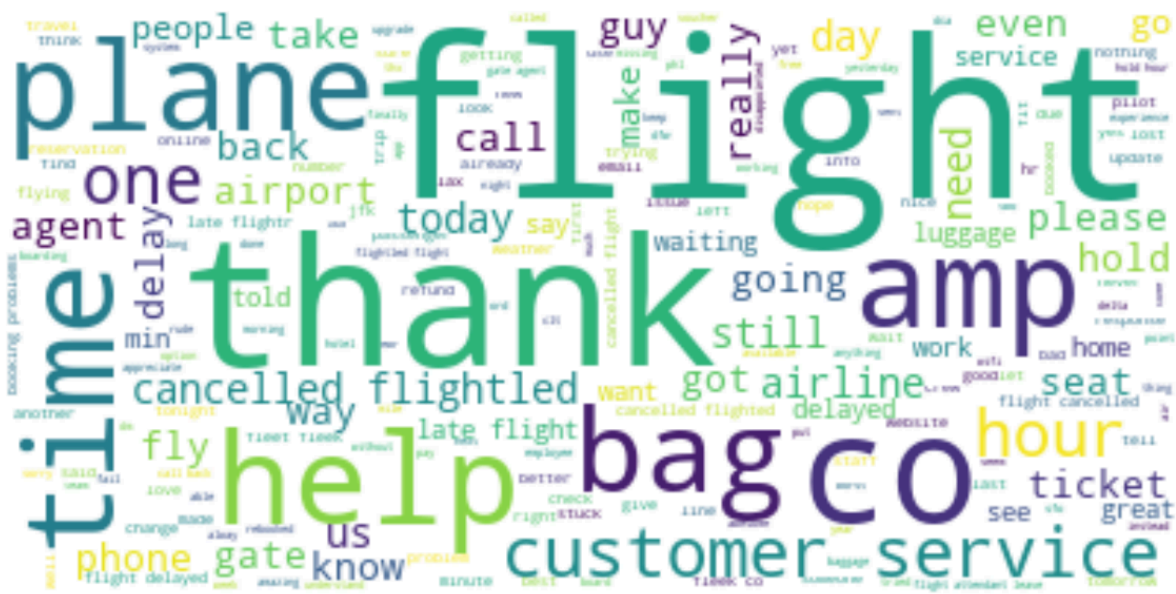
```
Total number of words in Corpus are  143680
```

Word cloud

In [180]:
```
text = " ".join(review for review in new_data['clean_text'])
print ("There are {} words in the combination of all review.".format(l
en(text)))
```

```
There are 918374 words in the combination of all review.
```

In [181]:
```
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(text)
plt.figure(figsize = (20,20))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

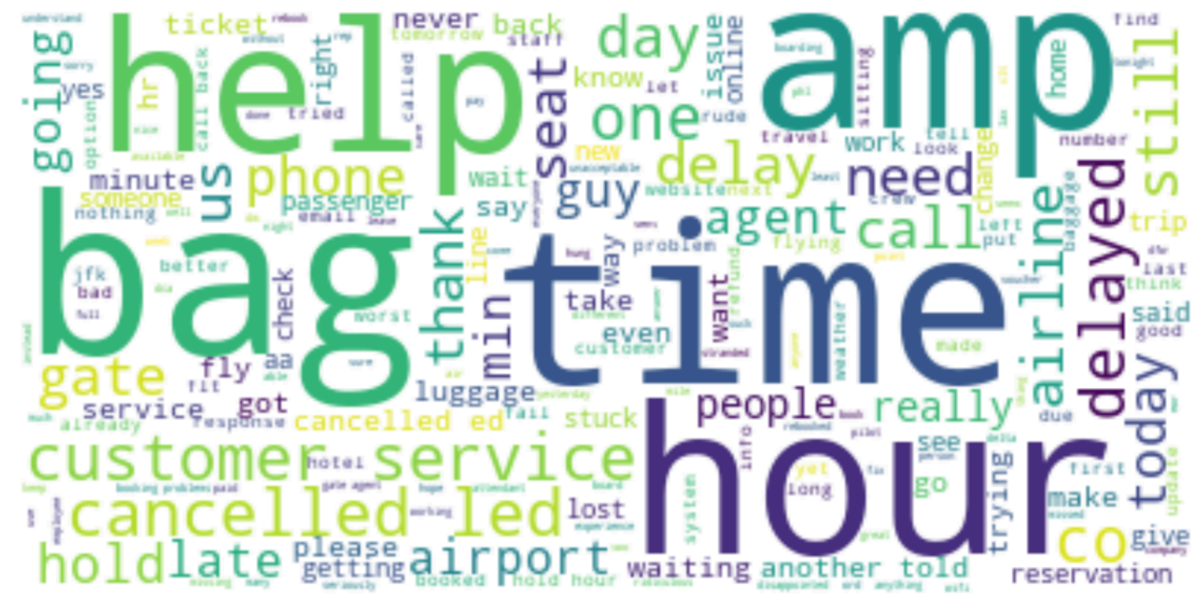


Observations:

- flight and thank is the most common used words. We can remove flight since it is a common word

Wordcloud for records with negative sentiment

In [182]:
```
textNeg = " ".join(review for review in new_data['clean_text'].loc[new
_data['airline_sentiment']==1])
textNeg=textNeg.replace('flight', '')
textNeg=textNeg.replace('plane', '')
wordcloud = WordCloud(background_color="white").generate(textNeg)
plt.figure(figsize = (20,20))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



Observations:

- People are complaining about bag(maybe they would have lost their bags), time, delayed, cancelled and customer service.

Wordcloud for records with positive sentiment

In [185]:
```
textPos = " ".join(review for review in new_data['clean_text'].loc[new
_data['airline_sentiment']==0])
textPos=textPos.replace('flight', '')
wordcloud = WordCloud(background_color="white").generate(textPos)
plt.figure(figsize = (20,20))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Observations:

- People like to say thank you and great when they are happy with airline.

In [186]:
```
new_data["airline_sentiment"].value_counts()
```

Out[186]:
```
1    9178
0    5462
Name: airline_sentiment, dtype: int64
```

In [187]:
```
new_data.loc[new_data['clean_text'].isnull()].shape
```

Out[187]:
```
(0, 2)
```

In [188]:
```
#Removing records with Null values
new_data=new_data.loc[new_data['clean_text'].notnull()]
```

In [189]:
```
new_data.shape
```

Out[189]:
```
(14640, 2)
```

Vectorization

In [194]:
```
# Vectorization (Convert text data to numbers).
from sklearn.feature_extraction.text import CountVectorizer

bow_vec = CountVectorizer(max_features=20000)                # Keep on
ly 2000 features as number of features will increase the processing ti
me.
data_features = bow_vec.fit_transform(new_data['clean_text'])

data_features = data_features.toarray()
```

In [195]:
```
data_features.shape
```

Out[195]:
```
(14640, 14917)
```

In [196]:
```
labels = new_data['airline_sentiment']
labels = labels.astype('int')
```

In [197]:
```
# Split data into training and testing set.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data_features, lab
els, test_size=0.3, random_state=42)
```

In [198]:
```
# Using Random Forest to build model for the classification of review
s.
# Also calculating the cross validation score.

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score

forest = RandomForestClassifier(n_estimators=10, n_jobs=4)

forest = forest.fit(X_train, y_train)

print(forest)

print(np.mean(cross_val_score(forest, data_features, labels, cv=10)))
```

```
RandomForestClassifier(n_estimators=10, n_jobs=4)
0.7871584699453552
```

Observations:

- We got 79% of accuracy in the training data

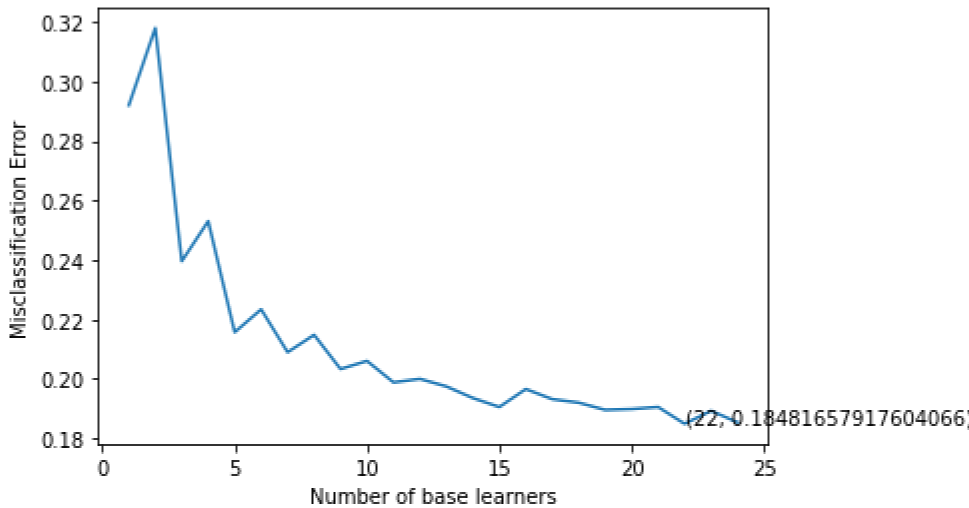
Optimizing the parameter: Number of trees in the random forest model(n_estimators)

In [199]:
```
# Finding optimal number of base learners using k-fold CV ->
base_ln = [x for x in range(1, 25)]
base_ln
```

Out[199]:
```
[1,
 2,
 3,
 4,
 5,
 6,
 7,
 8,
 9,
 10,
 11,
 12,
 13,
 14,
 15,
 16,
 17,
 18,
 19,
 20,
 21,
 22,
 23,
 24]
```

In [200]:
```
# K-Fold Cross - validation .
cv_scores = []
for b in base_ln:
    clf = RandomForestClassifier(n_estimators = b)
    scores = cross_val_score(clf, X_train, y_train, cv = 5, scoring =
'accuracy')
    cv_scores.append(scores.mean())
```

In [201]:
```
# plotting the error as k increases
error = [1 - x for x in cv_scores]                                    #er
ror corresponds to each nu of estimator
optimal_learners = base_ln[error.index(min(error))]                   #Se
lection of optimal nu of n_estimator corresponds to minimum error.
plt.plot(base_ln, error)                                              #Pl
ot between each nu of estimator and misclassification error
xy = (optimal_learners, min(error))
plt.annotate('(%s, %s)' % xy, xy = xy, textcoords='data')
plt.xlabel("Number of base learners")
plt.ylabel("Misclassification Error")
plt.show()
```



In [202]:
```
# Training the best model and calculating accuracy on test data .
clf = RandomForestClassifier(n_estimators = optimal_learners)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

Out[202]: 0.8203551912568307

Observations:

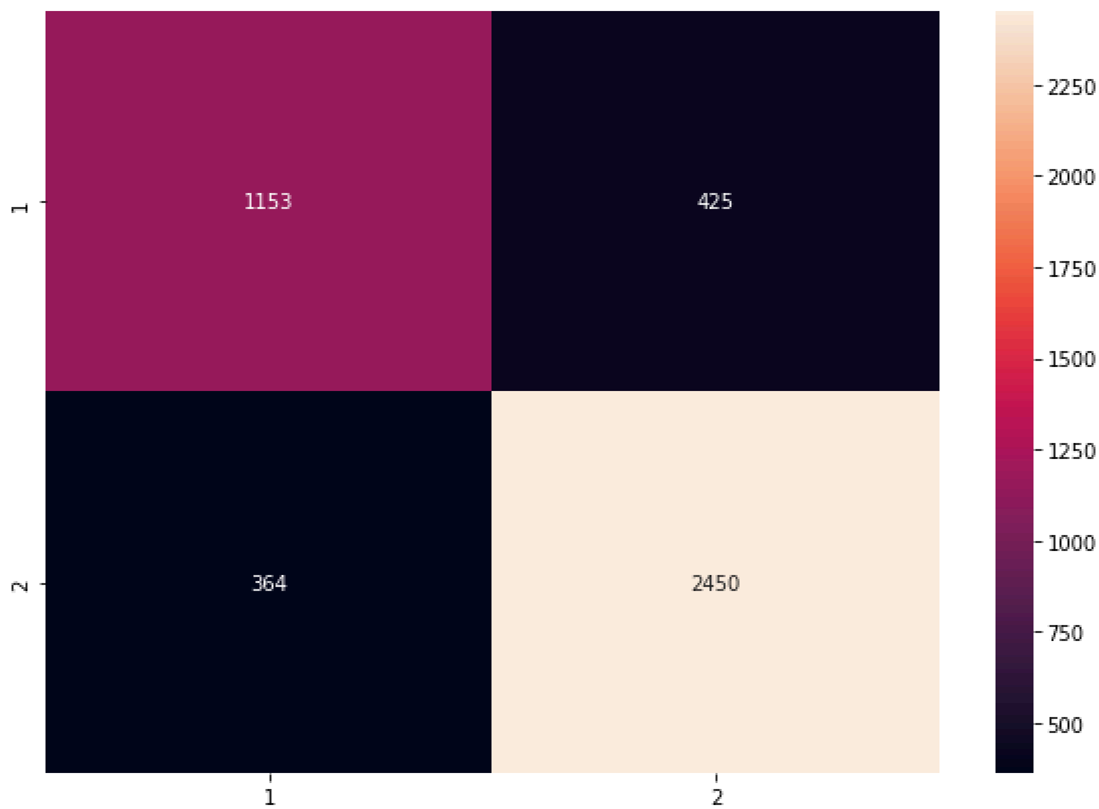
- We got 82% of accuracy in the test data

In [203]:
```
result =  clf.predict(X_test)                         #saving the prediction
on test data as a result
```

In [207]:
```
# Print and plot Confusion matirx to get an idea of how the distributi
on of the prediction is, among all the classes.

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.metrics import confusion_matrix

conf_mat = confusion_matrix(y_test, result)

print(conf_mat)

print(metrics.f1_score(y_test, result,average='micro'))

df_cm = pd.DataFrame(conf_mat, index = [i for i in "12"],
                     columns = [i for i in "12"])
plt.figure(figsize = (10,7))
sns.heatmap(df_cm, annot=True, fmt='g')
```

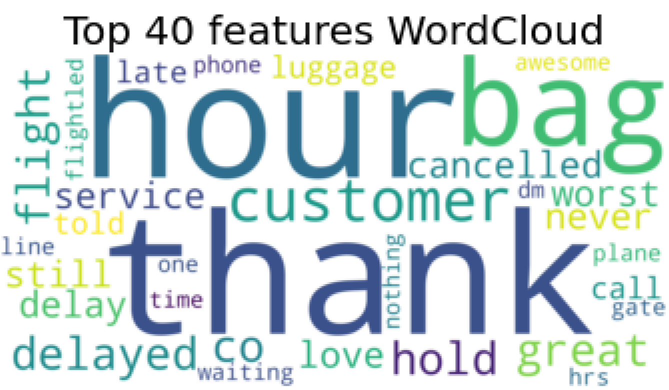
```
[[1153  425]
 [ 364 2450]]
0.8203551912568307
```

Out[207]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9ab792c510>



Word Cloud of top 40 important features from the CountVectorizer + Random Forest based model

In [208]:
```
all_features = bow_vec.get_feature_names()                #Instantiate t
he feature from the vectorizer
top_features=''                                              # Addition
of top 40 feature into top_feature after training the model
feat=clf.feature_importances_
features=np.argsort(feat)[::-1]
for i in features[0:40]:
    top_features+=all_features[i]
    top_features+=' '



from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",colormap='viridis',widt
h=2000,
                          height=1000).generate(top_features)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.figure(1, figsize=(14, 11), frameon='equal')
plt.title('Top 40 features WordCloud', fontsize=20)
plt.axis("off")
plt.show()
```



Observations:

- hour and thank are the two most used feature among the top 40 features.

Term Frequency(TF) - Inverse Document Frequency(IDF)

In [211]:
```
# Using TfidfVectorizer to convert text data to numbers.

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=20000)
data_features = vectorizer.fit_transform(new_data['clean_text'])

data_features = data_features.toarray()

data_features.shape
```

Out[211]:
```
(14640, 14917)
```

In [212]:
```
# Split data into training and testing set.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data_features, lab
els, test_size=0.3, random_state=42)
```

In [213]:
```
# Using Random Forest to build model for the classification of review
s.
# Also calculating the cross validation score.

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score

import numpy as np

forest = RandomForestClassifier(n_estimators=10, n_jobs=4)

forest = forest.fit(X_train, y_train)

print(forest)

print(np.mean(cross_val_score(forest, data_features, labels, cv=5)))
```

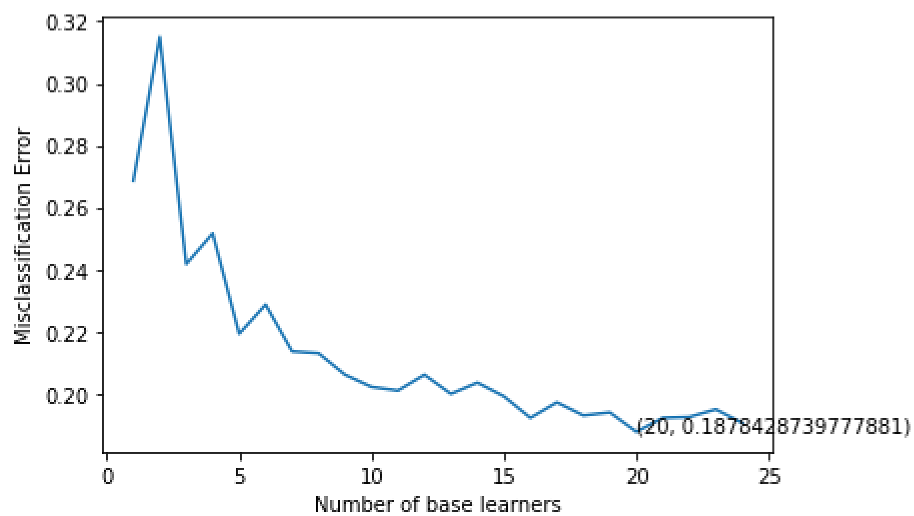
```
RandomForestClassifier(n_estimators=10, n_jobs=4)
0.798224043715847
```

Observations:

- We got 80% of the accuracy in the training data

In [214]:
```
# K - Fold Cross Validation .
cv_scores = []
for b in base_ln:
    clf = RandomForestClassifier(n_estimators = b)
    scores = cross_val_score(clf, X_train, y_train, cv = 5, scoring = 
'accuracy')
    cv_scores.append(scores.mean())
```

In [215]:
```
# plotting the error as k increases
error = [1 - x for x in cv_scores]
#error corresponds to each nu of estimator
optimal_learners = base_ln[error.index(min(error))]
#Selection of optimal nu of n_estimator corresponds to minimum error.
plt.plot(base_ln, error)
#Plot between each nu of estimator and misclassification error
xy = (optimal_learners, min(error))
plt.annotate('(%s, %s)' % xy, xy = xy, textcoords='data')
plt.xlabel("Number of base learners")
plt.ylabel("Misclassification Error")
plt.show()
```



In [216]:
```
# Training the best model and calculating error on test data .
clf = RandomForestClassifier(n_estimators = optimal_learners)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

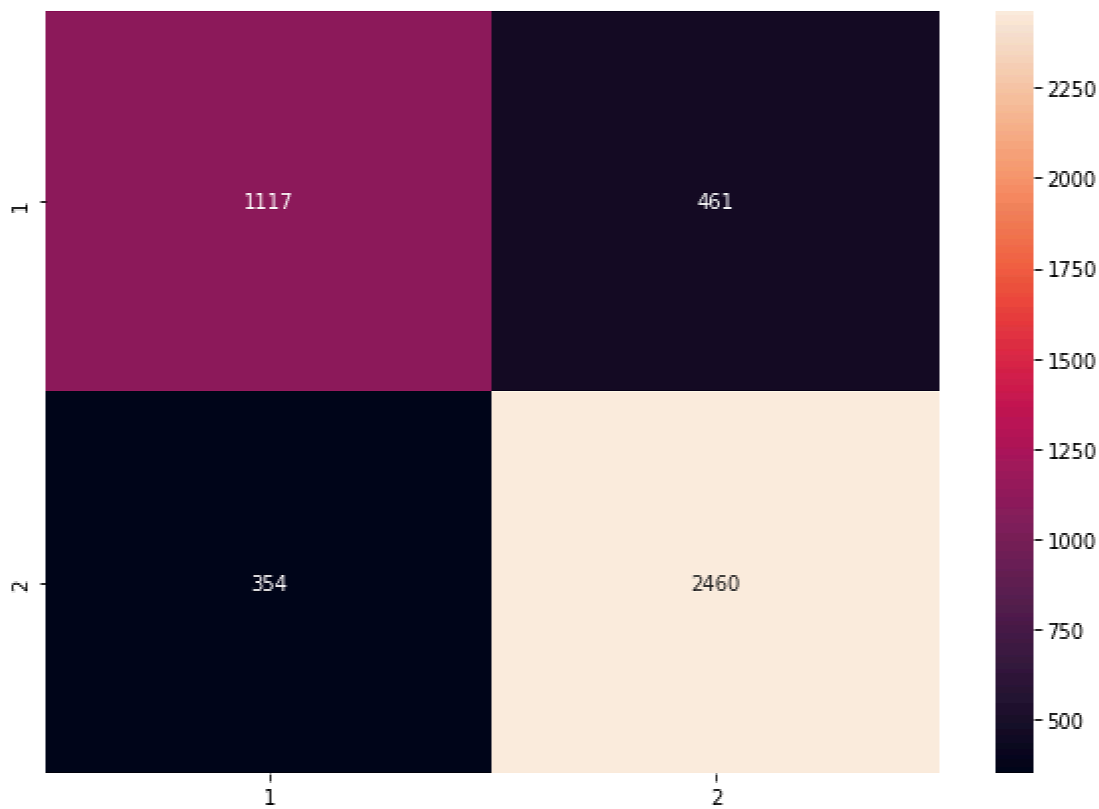
Out[216]: 0.8144353369763205

In [217]:
```
result = clf.predict(X_test)
```

In [218]:
```
# Print and plot Confusion matirx to get an idea of how the distributi
on of the prediction is, among all the classes.
result =  clf.predict(X_test)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.metrics import confusion_matrix

conf_mat = confusion_matrix(y_test, result)

print(conf_mat)

print(metrics.f1_score(y_test, result,average='micro'))

df_cm = pd.DataFrame(conf_mat, index = [i for i in "12"],
                  columns = [i for i in "12"])
plt.figure(figsize = (10,7))
sns.heatmap(df_cm, annot=True, fmt='g')
```

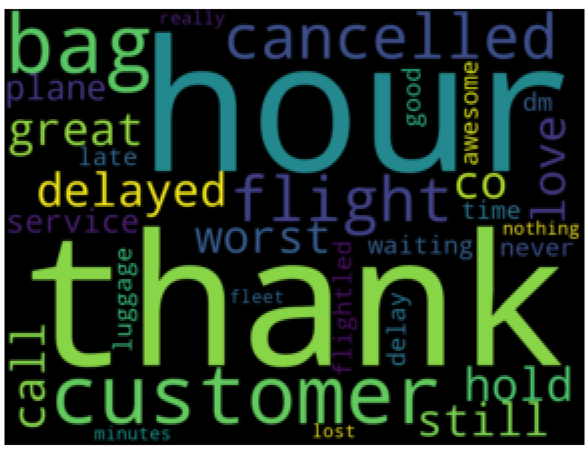
```
[[1117  461]
 [ 354 2460]]
0.8144353369763205
```

Out[218]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f9ab7702f90>`



In [219]:
```
all_features = vectorizer.get_feature_names()
#Instantiate the feature from the vectorizer
Top_features=''
#Addition of top 40 feature into top_feature after training the model
feat=clf.feature_importances_
features=np.argsort(feat)[::-1]
for i in features[0:40]:
    Top_features+=all_features[i]
    Top_features+=' '


from wordcloud import WordCloud
wordcloud = WordCloud(background_color="Black",width=1000,
                      height=750).generate(Top_features)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.figure(1, figsize=(30, 30), frameon='equal')
plt.title('Top 40 features WordCloud', fontsize=30)
plt.axis("off")
plt.show()
```

Top 40 features WordCloud



Observations:

- hour(negative sentiment) and thank(positive sentiment) are the two most used feature among the top 40 features.

Summary

- We used a dataset which has airlines reviews in text format and their sentiment positive, neutral and negative.
- The goal was to build a model for text-classification.
- We created the sentiment and usefulness column based on the helpfulness column.
- We pre-processed the data using various techniques and libraries.
- We created a Word Cloud plot based on summary and high and low score.
- The pre-processed data is converted to numbers (vectorized), so that we can feed the data into the model.
- We trained the model and optimized the parameter, which led to an increase the overall accuracy.
- After building the classification model, we predicted the results for the test data.
- We saw that using the above techniques, our model performed well in perspective of how text classification models perform.
- We can apply other model tuning and hyperparameter tuning techniques, as well as other pre-processing techniques to increase the overall accuracy even further.