Data Preprocessing

In [1]:

```
# Package imports
import pandas as pd
import numpy as np
from time import time
from bs4 import BeautifulSoup
import spacy
!pip install unidecode
!pip install word2number
import unidecode
from word2number import w2n
!pip install contractions
import contractions

# Loading the data into pandas dataframe
tweetsdf = pd.read_csv(r"C:\Users\srini\OneDrive\Desktop\Advanced Machine Lea
tweetsdf.info()
```

```
Requirement already satisfied: unidecode in c:\users\srini\anaconda3\lib\si
te-packages (1.3.4)
Requirement already satisfied: word2number in c:\users\srini\anaconda3\lib
\site-packages (1.1)
Requirement already satisfied: contractions in c:\users\srini\anaconda3\lib
\site-packages (0.1.68)
Requirement already satisfied: textsearch>=0.0.21 in c:\users\srini\anacond
a3\lib\site-packages (from contractions) (0.0.21)
Requirement already satisfied: anyascii in c:\users\srini\anaconda3\lib\sit
e-packages (from textsearch>=0.0.21->contractions) (0.3.1)
Requirement already satisfied: pyahocorasick in c:\users\srini\anaconda3\li
b\site-packages (from textsearch>=0.0.21->contractions) (1.4.4)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   tweet_id                    14640 non-null  float64
 1   airline_sentiment           14640 non-null  object
 2   airline_sentiment_confidence 14640 non-null  float64
 3   negativereason              9178 non-null   object
 4   negativereason_confidence   10522 non-null  float64
 5   airline                     14640 non-null  object
 6   airline_sentiment_gold      40 non-null     object
 7   name                        14640 non-null  object
 8   negativereason_gold         32 non-null     object
 9   retweet_count               14640 non-null  int64
 10  text                        14640 non-null  object
 11  tweet_coord                 1019 non-null   object
 12  tweet_created               14640 non-null  object
 13  tweet_location              9907 non-null   object
 14  user_timezone               9820 non-null   object
dtypes: float64(3), int64(1), object(11)
memory usage: 1.7+ MB
```

In [2]:
```
tweetsdf.head()
```

Out[2]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereaso
0	5.700000e+17	neutral	1.0000	NaN	
1	5.700000e+17	positive	0.3486	NaN	
2	5.700000e+17	neutral	0.6837	NaN	
3	5.700000e+17	negative	1.0000	Bad Flight	
4	5.700000e+17	negative	1.0000	Can't Tell	

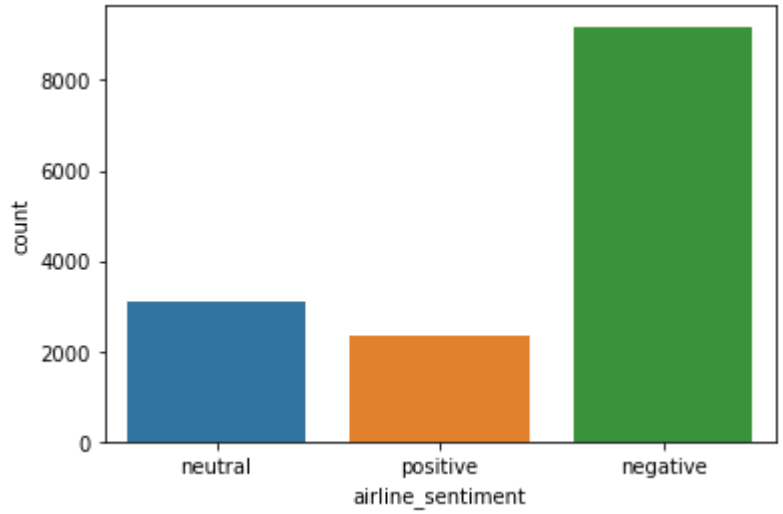
In [3]:
```
# Drop columns
tweetsdf.drop(columns = ['tweet_id', 'airline_sentiment_gold', 'negativereasc
```

In [4]:
```
# NUmber of each sentiment reviews
print('Number of negative tweets:', tweetsdf[tweetsdf['airline_sentiment'] ==
print('Number of positive tweets:', tweetsdf[tweetsdf['airline_sentiment'] ==
print('Number of neutral tweets:', tweetsdf[tweetsdf['airline_sentiment'] ==
```

```
Number of negative tweets: 9178
Number of positive tweets: 2363
Number of neutral tweets: 3099
```

In [5]:
```
import seaborn as sns
sns.countplot(x = "airline_sentiment", data = tweetsdf)
```

Out[5]: <AxesSubplot:xlabel='airline_sentiment', ylabel='count'>



In [6]:
```
# Replacing 'neutral' & 'positive' with 'non-negative' respectively
tweetsdf['airline_sentiment'].replace('positive', 'non-negative', inplace=Tru
tweetsdf['airline_sentiment'].replace('neutral', 'non-negative', inplace=True
tweetsdf.head()
```

Out[6]:

	airline_sentiment	text
0	non-negative	@VirginAmerica What @dhepburn said.
1	non-negative	@VirginAmerica plus you've added commercials t...
2	non-negative	@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 [7]:
```
# Finding the duplicate values
tweetsdf.duplicated().sum()
```

Out[7]: 205

In [8]:
```
# Dropping duplicates
tweetsdf = tweetsdf.drop_duplicates(keep='first')
tweetsdf.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
# Checking for any null values
tweetsdf.isnull().all()
```

Out[9]:
```
airline_sentiment    False
text                 False
dtype: bool
```

In [10]:
```
tweetsdf.head()
```

Out[10]:

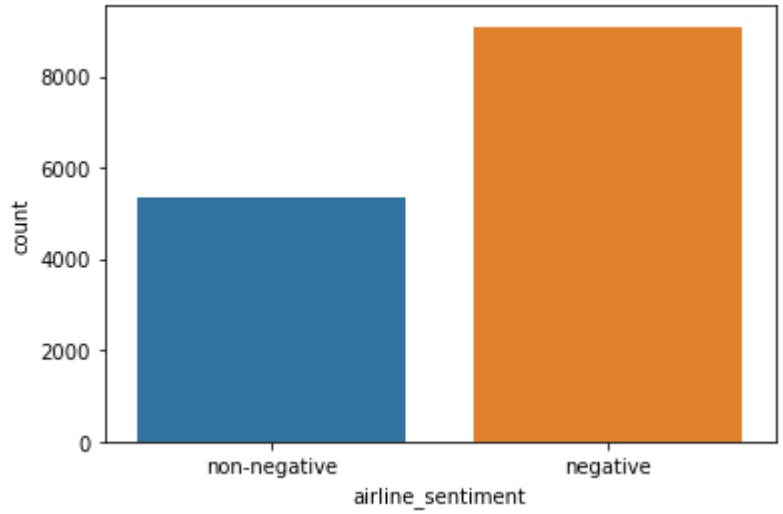
	airline_sentiment	text
0	non-negative	@VirginAmerica What @dhepburn said.
1	non-negative	@VirginAmerica plus you've added commercials t...
2	non-negative	@VirginAmerica I didn't today... Must mean I n...
3	negative	@VirginAmerica it's really aggressive to blast...
4	negative	@VirginAmerica and it's a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

In [11]:
```
tweetsdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14435 entries, 0 to 14639
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   airline_sentiment  14435 non-null  object
 1   text               14435 non-null  object
dtypes: object(2)
memory usage: 338.3+ KB
```

In [12]:
```
import seaborn as sns
sns.countplot(x = "airline_sentiment", data = tweetsdf)
```

Out[12]: <AxesSubplot:xlabel='airline_sentiment', ylabel='count'>



In [13]:
```
tweetsdf.text[73]
```

Out[13]: '@VirginAmerica your airline is awesome but your lax loft needs to step up its game. $40 for dirty tables and floors? http://t.co/hy0VrfhjHt' (http://t.co/hy0VrfhjHt')

In [14]:
```
tweetsdf.text[233]
```

Out[14]: "@VirginAmerica, the only airline based in Silicon Valley! #disruption #FCmostinnovative #incubator @FastCompany's http://t.co/wU3LbCNcr9" (http://t.co/wU3LbCNcr9")

In [15]:
```
tweetsdf.text[24]
```

Out[15]: '@VirginAmerica you guys messed up my seating.. I reserved seating with my friends and you guys gave my seat away ... 😡 I want free internet'

In [16]:

```
# Remove any URL's from the data
import re
def remove_URL(sample):
    """Remove URLs from a sample string"""
    return re.sub(r"http\S+", "", sample)

tweetsdf['no_URL'] = tweetsdf['text'].apply(lambda x: [remove_URL(word) for w
tweetsdf.head()
```

Out[16]:

	airline_sentiment	text	no_URL
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...
3	negative	@VirginAmerica it's really aggressive to blast...	[@VirginAmerica, it's, really, aggressive, to,...
4	negative	@VirginAmerica and it's a really big bad thing...	[@VirginAmerica, and, it's, a, really, big, ba...

In [17]:

```
tweetsdf.no_URL[73]
```

Out[17]:

```
['@VirginAmerica',
 'your',
 'airline',
 'is',
 'awesome',
 'but',
 'your',
 'lax',
 'loft',
 'needs',
 'to',
 'step',
 'up',
 'its',
 'game.',
 '$40',
 'for',
 'dirty',
 'tables',
 'and',
 'floors?',
 '']
```

In [18]:

```
# Remove any html tags from the data
def strip_html_tags(text):
    soup = BeautifulSoup(text, "html.parser")
    stripped_text = soup.get_text(separator=" ")
    return stripped_text

tweetsdf['no_html'] = tweetsdf['no_URL'].apply(lambda x: [strip_html_tags(wor
tweetsdf.head()
```

```
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "..." looks like a filename, not markup. You should pro
bably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "." looks like a filename, not markup. You should proba
bly open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "/" looks like a filename, not markup. You should proba
bly open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: ".." looks like a filename, not markup. You should prob
ably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "...." looks like a filename, not markup. You should pr
obably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "con" looks like a filename, not markup. You should pro
bably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "images" looks like a filename, not markup. You should
probably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "data" looks like a filename, not markup. You should pr
obably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "Data" looks like a filename, not markup. You should pr
obably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "....." looks like a filename, not markup. You should p
robably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "test" looks like a filename, not markup. You should pr
obably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "TEST" looks like a filename, not markup. You should pr
```

```
obably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "............" looks like a filename, not markup. You
should probably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "data." looks like a filename, not markup. You should p
robably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "......." looks like a filename, not markup. You should
probably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
C:\Users\srini\anaconda3\lib\site-packages\bs4\__init__.py:332: MarkupResem
blesLocatorWarning: "......" looks like a filename, not markup. You should
probably open this file and pass the filehandle into Beautiful Soup.
  warnings.warn(
```

Out[18]:

	airline_sentiment	text	no_URL	no_html
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...
3	negative	@VirginAmerica it's really aggressive to blast...	[@VirginAmerica, it's, really, aggressive, to,...	[@VirginAmerica, it's, really, aggressive, to,...
4	negative	@VirginAmerica and it's a really big bad thing...	[@VirginAmerica, and, it's, a, really, big, ba...	[@VirginAmerica, and, it's, a, really, big, ba...

In [19]:

```
# Remove accented characters
def remove_accented_chars(text):
    """remove accented characters from text, e.g. café"""
    text = unidecode.unidecode(text)
    return text

tweetsdf['no_accentchar'] = tweetsdf['no_html'].apply(lambda x: [unidecode.un
tweetsdf.head()
```

Out[19]:

	airline_sentiment	text	no_URL	no_html	no_accentchar
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...
3	negative	@VirginAmerica it's really aggressive to blast...	[@VirginAmerica, it's, really, aggressive, to,...	[@VirginAmerica, it's, really, aggressive, to,...	[@VirginAmerica, it's, really, aggressive, to,...
4	negative	@VirginAmerica and it's a really big bad thing...	[@VirginAmerica, and, it's, a, really, big, ba...	[@VirginAmerica, and, it's, a, really, big, ba...	[@VirginAmerica, and, it's, a, really, big, ba...

In [21]:
```
tweetsdf.no_accentchar[24]
```

Out[21]:
```
['@VirginAmerica',
 'you',
 'guys',
 'messed',
 'up',
 'my',
 'seating..',
 'I',
 'reserved',
 'seating',
 'with',
 'my',
 'friends',
 'and',
 'you',
 'guys',
 'gave',
 'my',
 'seat',
 'away',
 '...',
 '',
 'I',
 'want',
 'free',
 'internet']
```

In [22]:
```
tweetsdf.no_accentchar[18]
```

Out[22]:
```
['I', '', 'flying', '@VirginAmerica.', '']
```

In [23]:
```
# Expanding contractions 'you've to you have'
!pip install contractions
import contractions
def expand_contractions(text):
    text = contractions.fix(text)
    return text
```

```
Requirement already satisfied: contractions in c:\users\srini\anaconda3\lib
\site-packages (0.1.68)
Requirement already satisfied: textsearch>=0.0.21 in c:\users\srini\anacond
a3\lib\site-packages (from contractions) (0.0.21)
Requirement already satisfied: pyahocorasick in c:\users\srini\anaconda3\li
b\site-packages (from textsearch>=0.0.21->contractions) (1.4.4)
Requirement already satisfied: anyascii in c:\users\srini\anaconda3\lib\sit
e-packages (from textsearch>=0.0.21->contractions) (0.3.1)
```

In [25]:
```
tweetsdf['no_contract'] = tweetsdf['no_accentchar'].apply(lambda x: [contract
tweetsdf.head()
```

Out[25]:

	airline_sentiment	text	no_URL	no_html	no_accentchar	no
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@Virg @
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@Virg plus, added,
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@Virg tod
3	negative	@VirginAmerica it's really aggressive to blast...	[@VirginAmerica, it's, really, aggressive, to,...	[@VirginAmerica, it's, really, aggressive, to,...	[@VirginAmerica, it's, really, aggressive, to,...	[@Virg aggre
4	negative	@VirginAmerica and it's a really big bad thing...	[@VirginAmerica, and, it's, a, really, big, ba...	[@VirginAmerica, and, it's, a, really, big, ba...	[@VirginAmerica, and, it's, a, really, big, ba...	[@Virg a rea

In [26]:
```
tweetsdf['no_contract_str'] = [' '.join(map(str, l)) for l in tweetsdf['no_co
tweetsdf.head()
```

Out[26]:

	airline_sentiment	text	no_URL	no_html	no_accentchar	no
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@Virg @
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@Virg plus, added,
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@Virg tod
3	negative	@VirginAmerica it's really aggressive to blast...	[@VirginAmerica, it's, really, aggressive, to,...	[@VirginAmerica, it's, really, aggressive, to,...	[@VirginAmerica, it's, really, aggressive, to,...	[@Virg aggre
4	negative	@VirginAmerica and it's a really big bad thing...	[@VirginAmerica, and, it's, a, really, big, ba...	[@VirginAmerica, and, it's, a, really, big, ba...	[@VirginAmerica, and, it's, a, really, big, ba...	[@Virg a rea

In [27]:

```
# Remove punctuations
import string
def remove_punct(text):
    text_nonpunct = "".join([char for char in text if char not in string.punc
    return text_nonpunct

tweetsdf['no_punc_text'] = tweetsdf['no_contract_str'].apply(lambda x: remove
tweetsdf.head()
```

Out[27]:

	airline_sentiment	text	no_URL	no_html	no_accentchar	
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@ ac
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@
3	negative	@VirginAmerica it's really aggressive to	[@VirginAmerica, it's, really, aggressive, to	[@VirginAmerica, it's, really, aggressive, to	[@VirginAmerica, it's, really, aggressive, to	[@

In [29]:

```
tweetsdf.no_punc_text[18]
```

Out[29]: 'I flying VirginAmerica '

In [30]:

```
# Tokenization of the data
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer
tweetsdf['tokenized'] = tweetsdf['no_punc_text'].apply(word_tokenize)
tweetsdf.head()
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\srini\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[30]:

	airline_sentiment	text	no_URL	no_html	no_accentchar	
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@ ac
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@

In [31]:

```
tweetsdf.tokenized[18]
```

Out[31]:

```
['I', 'flying', 'VirginAmerica']
```

In [32]:
```
# convert text to lower case
tweetsdf['lower'] = tweetsdf['tokenized'].apply(lambda x: [word.lower() for w
tweetsdf.head()
```

Out[32]:

	airline_sentiment	text	no_URL	no_html	no_accentchar	
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@ ac
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@
3	negative	@VirginAmerica it's really aggressive to	[@VirginAmerica, it's, really, aggressive, to	[@VirginAmerica, it's, really, aggressive, to	[@VirginAmerica, it's, really, aggressive, to	[@

In [33]:
```
tweetsdf.lower[18]
```

Out[33]:
```
['i', 'flying', 'virginamerica']
```

In [36]:
```
# remove stop words
nltk_stopwords = nltk.corpus.stopwords.words('english')
nltk_stopwords.remove('but')
nltk_stopwords.remove('no')
nltk_stopwords.remove('not')

stop_words = set(stopwords.words('english'))
tweetsdf['no_stopwords'] = tweetsdf['lower'].apply(lambda x: [word for word i
tweetsdf.head()
```

Out[36]:

	airline_sentiment	text	no_URL	no_html	no_accentchar	
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@ ad
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@
3	negative	@VirginAmerica it's really aggressive to	[@VirginAmerica, it's, really, aggressive, to	[@VirginAmerica, it's, really, aggressive, to	[@VirginAmerica, it's, really, aggressive, to	[@

In [38]:
```
tweetsdf.no_stopwords[18]
```

Out[38]:
```
['flying', 'virginamerica']
```

In [39]:
```
tweetsdf.no_stopwords[328]
```

Out[39]:
```
['virginamerica',
 'shrinerack',
 'seattle',
 'bound',
 'wifey',
 'got',
 'duffle',
 'vday',
 'keeper',
 'holla']
```

In [40]:
```
# parts of speech of each word for lemmatization purpose
nltk.download('averaged_perceptron_tagger')
tweetsdf['pos_tags'] = tweetsdf['no_stopwords'].apply(nltk.tag.pos_tag)
tweetsdf.head()
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\srini\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[40]:

	airline_sentiment	text	no_URL	no_html	no_accentchar	
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@ ac
2	non-negative	@VirginAmerica I didn't today	[@VirginAmerica, I, didn't, today	[@VirginAmerica, I, didn't, today	[@VirginAmerica, I, didn't, today	[@

In [41]:
```
tweetsdf.pos_tags[18]
```

Out[41]:
```
[('flying', 'VBG'), ('virginamerica', 'NN')]
```

In [42]:

```
nltk.download('wordnet')
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
tweetsdf['wordnet_pos'] = tweetsdf['pos_tags'].apply(lambda x: [(word, get_wo
tweetsdf.head()
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\srini\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[42]:

	airline_sentiment	text	no_URL	no_html	no_accentchar	no
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@Virg @
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@Virg plus, added,
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@Virg tod
3	negative	@VirginAmerica it's really aggressive to blast...	[@VirginAmerica, it's, really, aggressive, to,...	[@VirginAmerica, it's, really, aggressive, to,...	[@VirginAmerica, it's, really, aggressive, to,...	[@Virg aggre
4	negative	@VirginAmerica and it's a really big bad thing...	[@VirginAmerica, and, it's, a, really, big, ba...	[@VirginAmerica, and, it's, a, really, big, ba...	[@VirginAmerica, and, it's, a, really, big, ba...	[@Virg a rea

In [43]:

```
tweetsdf.wordnet_pos[18]
```

Out[43]:

```
[('flying', 'v'), ('virginamerica', 'n')]
```

In [44]:
```
# Lemmatization of data
wnl = WordNetLemmatizer()
tweetsdf['lemmatized'] = tweetsdf['wordnet_pos'].apply(lambda x: [wnl.lemmati
tweetsdf.head()
```

Out[44]:

	airline_sentiment	text	no_URL	no_html	no_accentchar	
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@ ac
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@
3	negative	@VirginAmerica it's really aggressive to	[@VirginAmerica, it's, really, aggressive, to	[@VirginAmerica, it's, really, aggressive, to	[@VirginAmerica, it's, really, aggressive, to	[@

In [45]:
```
tweetsdf.lemmatized[18]
```

Out[45]:
```
['fly', 'virginamerica']
```

In [46]:
```
tweetsdf['lemmatized_str'] = [' '.join(map(str, l)) for l in tweetsdf['lemmat
tweetsdf.head()
```

Out[46]:

	airline_sentiment	text	no_URL	no_html	no_accentchar	
0	non-negative	@VirginAmerica What @dhepburn said.	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@VirginAmerica, What, @dhepburn, said.]	[@
1	non-negative	@VirginAmerica plus you've added commercials t...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@VirginAmerica, plus, you've, added, commerci...	[@ ac
2	non-negative	@VirginAmerica I didn't today... Must mean I n...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@VirginAmerica, I, didn't, today..., Must, me...	[@
3	negative	@VirginAmerica it's really aggressive to	[@VirginAmerica, it's, really, aggressive, to	[@VirginAmerica, it's, really, aggressive, to	[@VirginAmerica, it's, really, aggressive, to	[@

In [47]:
```
tweetsdf.lemmatized_str[18]
```

Out[47]:
```
'fly virginamerica'
```

In [49]:
```
# Drop columns
tweetsdf.drop(columns = ['text', 'no_URL', 'no_html', 'no_accentchar', 'no_cc
```

In [50]:
```
tweetsdf.to_csv('Finaldf1.csv')
```

In [32]:
```
tweetsdf.head()
```

Out[32]:

	airline_sentiment	lemmatized_text_str
0	non-negative	virginamerica dhepburn say
1	non-negative	virginamerica plus added commercial experience...
2	non-negative	virginamerica today must mean need take anothe...
3	negative	virginamerica really aggressive blast obnoxiou...
4	negative	virginamerica really big bad thing

LSTM with epochs = 2 & Vec size 10000

In [17]:

```
# Package imports
import pandas as pd
import numpy as np
from time import time
from bs4 import BeautifulSoup
import spacy
!pip install unidecode
!pip install word2number
import unidecode
from word2number import w2n
!pip install contractions
import contractions

# Loading the data into pandas dataframe
tweetsdf = pd.read_csv(r"C:\Users\srini\OneDrive\Desktop\Advanced Machine Lea
tweetsdf.info()
```

```
Requirement already satisfied: unidecode in c:\users\srini\anaconda3\lib\si
te-packages (1.3.4)
Requirement already satisfied: word2number in c:\users\srini\anaconda3\lib
\site-packages (1.1)
Requirement already satisfied: contractions in c:\users\srini\anaconda3\lib
\site-packages (0.1.68)
Requirement already satisfied: textsearch>=0.0.21 in c:\users\srini\anacond
a3\lib\site-packages (from contractions) (0.0.21)
Requirement already satisfied: pyahocorasick in c:\users\srini\anaconda3\li
b\site-packages (from textsearch>=0.0.21->contractions) (1.4.4)
Requirement already satisfied: anyascii in c:\users\srini\anaconda3\lib\sit
e-packages (from textsearch>=0.0.21->contractions) (0.3.1)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14435 entries, 0 to 14434
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         14435 non-null  int64
 1   airline_sentiment  14435 non-null  object
 2   lemmatized_str     14435 non-null  object
dtypes: int64(1), object(2)
memory usage: 338.4+ KB
```

In [18]:

```
tweetsdf.drop(columns = ['Unnamed: 0' ], axis = 1, inplace = True)
```

In [19]:

```
# method to tokenize documents with lemmatization
import nltk
import string
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer

def lemma_tokenizer(corpus):   # a method to lemmatize corpus
    corpus = ''.join([ch for ch in corpus if ch not in string.punctuation])
    tokens = nltk.word_tokenize(corpus)
    lemmatizer = WordNetLemmatizer()
    return [lemmatizer.lemmatize(token) for token in tokens]

nltk_stopwords = nltk.corpus.stopwords.words('english')

# construct term-document (tf) matrix with lemmatization - not using data yet
tf = CountVectorizer(tokenizer=lemma_tokenizer,  # use lemma_tokenizer
                     stop_words=nltk_stopwords,  # use customized stopwords l
                     ngram_range=(1,2))          # use unigrams and bigrams
```

In [4]:

```
# method to tokenize documents with lemmatization
import nltk
import string
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer

def lemma_tokenizer(corpus):   # a method to lemmatize corpus
    corpus = ''.join([ch for ch in corpus if ch not in string.punctuation])
    tokens = nltk.word_tokenize(corpus)
    lemmatizer = WordNetLemmatizer()
    return [lemmatizer.lemmatize(token) for token in tokens]

# use nltk stopwords list, but remove 'but', 'no' and 'not' from the stopwora

# construct term-document (tf) matrix with lemmatization - not using data yet
tf = CountVectorizer(tokenizer=lemma_tokenizer,  # use lemma_tokenizer
                     stop_words=nltk_stopwords,  # use customized stopwords l
                     ngram_range=(1,2))          # use unigrams and bigrams
```

In [5]:
```
# Split the data
from sklearn.model_selection import train_test_split
textdf = tweetsdf['lemmatized_str']
ydf = tweetsdf['airline_sentiment']
textdf_train, textdf_test, ydf_train, ydf_test = train_test_split(textdf, ydf
textdf_test[:5], ydf_test[:5]
```

Out[5]:
```
(3722     united fantastic job people today ua22 dublin ...
 14175    americanair kid one answer call reserv line du...
 5619     southwestairs ceo kelly draws record crowd bwi...
 7011                cmon bad jetbae rt jetblue fleet fleek
 1224     united thanks link finally arrive brussels 9 h...
 Name: lemmatized_str, dtype: object,
 3722     non-negative
 14175        negative
 5619     non-negative
 7011         negative
 1224         negative
 Name: airline_sentiment, dtype: object)
```

In [6]:
```
tf_train = tf.fit_transform(textdf_train)
tf_train
```

```
C:\Users\srini\anaconda3\lib\site-packages\sklearn\feature_extraction\text.
py:383: UserWarning: Your stop_words may be inconsistent with your preproce
ssing. Tokenizing the stop words generated tokens ['arent', 'couldnt', 'did
nt', 'doe', 'doesnt', 'dont', 'ha', 'hadnt', 'hasnt', 'havent', 'isnt', 'mi
ghtnt', 'mustnt', 'neednt', 'shant', 'shes', 'shouldnt', 'shouldve', 'thatl
l', 'wa', 'wasnt', 'werent', 'wont', 'wouldnt', 'youd', 'youll', 'youre',
'youve'] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
```

Out[6]:
```
<11548x80867 sparse matrix of type '<class 'numpy.int64'>'
        with 215965 stored elements in Compressed Sparse Row format>
```

In [7]:
```
tf_test = tf.transform(textdf_test)
tf_test
```

Out[7]:
```
<2887x80867 sparse matrix of type '<class 'numpy.int64'>'
        with 37950 stored elements in Compressed Sparse Row format>
```

In [8]:
```
# Sentiment analysis using LSTM with supervised word embeddings

# encode label (sentiment) into a numpy arry with 0/1 values (in alphabetical
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
label_encoder.fit(ydf)
y = label_encoder.transform(ydf)
print('Encoded class order:', label_encoder.classes_)
print('Before encoding:', label_encoder.inverse_transform(y)[0:5])
print('After encoding: ', y[0:5])     # 0 = +, 1 = -

# encode Sentiment into a dataframe with 0/1 values
# y = ydf.apply(lambda x: 1 if x=='+' else 0)  # 0 = -, 1 = +
# y.head()
```

```
Encoded class order: ['negative' 'non-negative']
Before encoding: ['non-negative' 'non-negative' 'non-negative' 'negative'
'negative']
After encoding:  [1 1 1 0 0]
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(textdf, y, test_size=0.2,
X_test[:5], y_test[:5]
```

Out[9]:
```
(3722     united fantastic job people today ua22 dublin ...
 14175    americanair kid one answer call reserv line du...
 5619     southwestairs ceo kelly draws record crowd bwi...
 7011                cmon bad jetbae rt jetblue fleet fleek
 1224     united thanks link finally arrive brussels 9 h...
 Name: lemmatized_str, dtype: object,
 array([1, 0, 1, 0, 0]))
```

In [10]:
```
# tokenize documents directly using tf.keras.preprocessing.text.Tokenizer
from tensorflow.keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer()   # tokenizer = Tokenizer(num_words=100)  # to use th
tokenizer.fit_on_texts(X_train)  # tokenize the documents and index the words
```

In [11]:
```
# transform words in documents to sequences of indexes, required by tf.keras.
print('X_train in text:\n', X_train[:5])
X_train = tokenizer.texts_to_sequences(X_train)
print('\nX_train after indexing:\n', X_train[:5])
```

```
X_train in text:
 2182     united flight cancel flightled 4 hr airport bo...
13688    americanair usairways food 4 hour flight 528 c...
11312    usairways well worthless regular service bunch...
8714     jetblue supervisor humiliate u uncompromising ...
9146     usairways really one book dividend mile travel...
Name: lemmatized_str, dtype: object

X_train after indexing:
 [[2, 1, 10, 41, 85, 100, 61, 42, 90, 31, 3212, 7, 120, 3213, 1, 3214, 144
5, 77], [4, 3, 442, 85, 8, 1, 4673, 1026, 709, 2547, 1158, 922, 16, 16],
[3, 95, 1885, 3215, 11, 1446, 153, 2548, 222], [6, 511, 2549, 19, 4674, 58
1, 4675], [3, 77, 31, 42, 643, 136, 74, 690, 24, 133, 3216, 2143, 3217]]
```

In [12]:
```
# tf.keras.layers.Embedding requires inputs to have equal length; so find max
max_len = np.max([len(doc) for doc in X_train])
max_len
```

Out[12]:
```
24
```

In [13]:
```
# pad shorter documents with zeros so that all documents have max_len
from tensorflow.keras.preprocessing.sequence import pad_sequences
X_train = pad_sequences(X_train, maxlen=max_len)  # padding='pre' (default) c
print('X_train after padding:\n', X_train)
```

```
X_train after padding:
 [[   0    0    0 ... 3214 1445   77]
 [   0    0    0 ...  922   16   16]
 [   0    0    0 ...  153 2548  222]
 ...
 [   0    0    0 ...   18  274  446]
 [   0    0    0 ... 2984  107   79]
 [   0    0    0 ...  182   14  125]]
```

In [14]:
```
print('X_test in text:\n', X_test[:5])
X_test = tokenizer.texts_to_sequences(X_test)
print('\nX_test after indexing:\n', X_test[:5])
```

```
X_test in text:
 3722     united fantastic job people today ua22 dublin ...
14175    americanair kid one answer call reserv line du...
5619     southwestairs ceo kelly draws record crowd bwi...
7011                cmon bad jetbae rt jetblue fleet fleek
1224     united thanks link finally arrive brussels 9 h...
Name: lemmatized_str, dtype: object

X_test after indexing:
 [[2, 1033, 286, 88, 47, 7099, 3322, 1620, 23, 104, 718], [4, 272, 31, 129,
17, 2330, 119, 101, 466, 17, 730, 78, 178, 22, 4101], [4319, 498, 2366, 835
6, 489, 2136, 3473, 204, 1865, 1560], [1247, 50, 4499, 266, 6, 151, 152],
[2, 9, 321, 183, 150, 551, 2495, 245]]
```

In [15]:
```
X_test = pad_sequences(X_test, maxlen=max_len)
print('X_test after padding:\n', X_test)
```

```
X_test after padding:
 [[   0    0    0 ...   23  104  718]
 [   0    0    0 ...  178   22 4101]
 [   0    0    0 ...  204 1865 1560]
 ...
 [   0    0    0 ...  620   34  390]
 [   0    0    0 ...  282    1    6]
 [   0    0    0 ...  486 1138 1494]]
```

In [16]:
```
vocab_size = len(tokenizer.word_index) + 1
vocab_size
```

Out[16]: 11874

In [16]:
```
# design LSTM model with Embedding layer (which must be the 1st layer)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SpatialDropout1D, Dropout
from tensorflow.keras.layers import LSTM, Dense
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

vec_size = 10000   # dimensionality of the word vectors

model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=vec_size, input_length=m
model.add(layers.LSTM(32, return_sequences=False))
model.add(Dense(1))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 24, 10000)         118740000

 lstm (LSTM)                 (None, 32)                1284224

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 120,024,257
Trainable params: 120,024,257
Non-trainable params: 0
_________________________________________________________________
```

In [17]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accurac
model.optimizer.get_config()
```

Out[17]:
```
{'name': 'Adam',
 'learning_rate': 0.001,
 'decay': 0.0,
 'beta_1': 0.9,
 'beta_2': 0.999,
 'epsilon': 1e-07,
 'amsgrad': False}
```

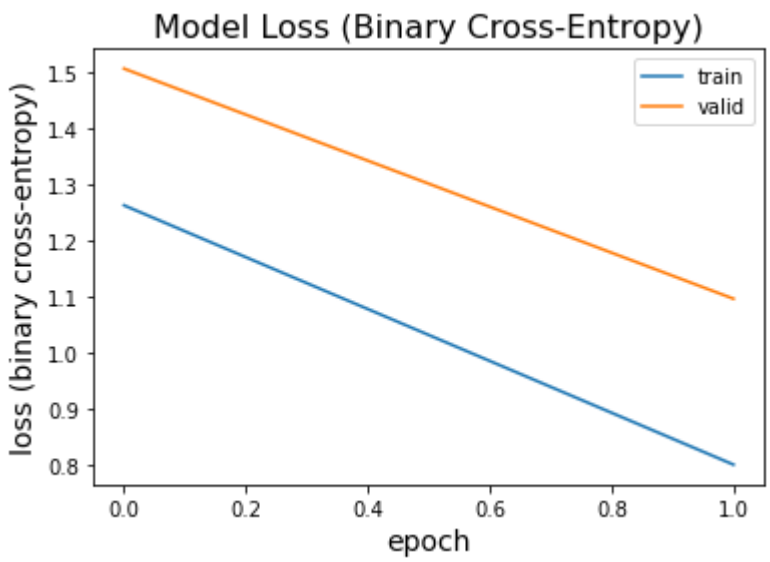
In [18]:

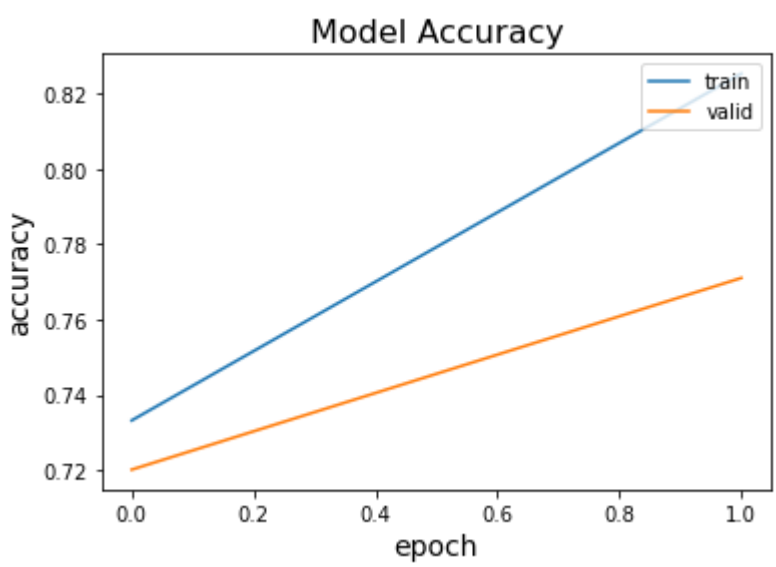
```
# train LSTM model
from time import time
t0 = time()
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), batch
print("\nTime to train LSTM model: %0.3f seconds." % (time() - t0))
```

```
Epoch 1/2
361/361 [==============================] - 924s 3s/step - loss: 1.2635 - ac
curacy: 0.7332 - val_loss: 1.5080 - val_accuracy: 0.7201
Epoch 2/2
361/361 [==============================] - 663s 2s/step - loss: 0.7999 - ac
curacy: 0.8253 - val_loss: 1.0965 - val_accuracy: 0.7710

Time to train LSTM model: 1587.165 seconds.
```

In [19]:

```
# plot training process
%matplotlib inline
import matplotlib.pyplot as plt

# plot for loss (mse)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss (Binary Cross-Entropy)', fontsize=16)
plt.xlabel('epoch', fontsize=14)
plt.ylabel('loss (binary cross-entropy)', fontsize=14)
plt.legend(['train', 'valid'], loc='upper right')
plt.show()

# plot for metrics
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy', fontsize=16)
plt.xlabel('epoch', fontsize=14)
plt.ylabel('accuracy', fontsize=14)
plt.legend(['train', 'valid'], loc='upper right')
plt.show()
```





In [20]:

```
# Method to perform evaluation for LSTM on test data
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix

def evaluate(model, y_test, y_pred, first_label, second_label, print_predict)

    if (print_predict==True):
        print('Prediction results:')
        print('Actual:   ', np.array(y_test))
        print('Predicted:', y_pred)

    print('\nTest accuracy:', accuracy_score(y_test, y_pred))
    print('\nConfusion matrix:\n', confusion_matrix(y_test, y_pred))

    cm = pd.DataFrame(confusion_matrix(y_test, y_pred, labels=[first_label, s
                      index=['actual ' + first_label, 'actual ' + second_labe
                      columns=['predicted ' + first_label, 'predicted ' + sec
    print('\nConfusion matrix with labels:\n', cm)
    print('\nLabel ' + first_label + ' is positive class')
    TP = cm.at['actual ' + first_label,  'predicted ' + first_label]
    FP = cm.at['actual ' + second_label, 'predicted ' + first_label]
    FN = cm.at['actual ' + first_label,  'predicted ' + second_label]
    TN = cm.at['actual ' + second_label, 'predicted ' + second_label]
    print('precision =', TP/(TP+FP), ', recall =', TP/(TP+FN), ', F1-score ='
```

In [21]:

```
y_pred_prob = model.predict(X_test)
# y_pred = (y_pred_prob > 0.5).astype('int32')    # if y_pred_prob>0.5 y_pred
y_pred_prob = pd.DataFrame(y_pred_prob, columns=['y_prob'])
y_pred = y_pred_prob['y_prob'].apply(lambda x: 'negative' if x < 0.5 else 'no
y_test = label_encoder.inverse_transform(y_test)
evaluate(model, y_test, np.array(y_pred), 'negative', 'non-negative', True)
```

```
Prediction results:
Actual:    ['non-negative' 'negative' 'non-negative' ... 'negative' 'negati
ve'
 'non-negative']
Predicted: ['non-negative' 'negative' 'non-negative' ... 'negative' 'negati
ve'
 'non-negative']

Test accuracy: 0.7710426047800485

Confusion matrix:
 [[1571  275]
 [ 386  655]]

Confusion matrix with labels:
                     predicted negative  predicted non-negative
actual negative                    1571                     275
actual non-negative                 386                     655

Label negative is positive class
precision = 0.8027593254982115 , recall = 0.8510292524377031 , F1-score =
0.8261898501183277 Specificity = 0.6292026897214217
```

LSTM with epochs = 3 & Vec size 10000

In [3]:

```
# Package imports
import pandas as pd
import numpy as np
from time import time
from bs4 import BeautifulSoup
import spacy
!pip install unidecode
!pip install word2number
import unidecode
from word2number import w2n
!pip install contractions
import contractions

# Loading the data into pandas dataframe
tweetsdf = pd.read_csv(r"C:\Users\srini\OneDrive\Desktop\Advanced Machine Lea
tweetsdf.info()
```

```
Requirement already satisfied: unidecode in c:\users\srini\anaconda3\lib\si
te-packages (1.3.4)
Requirement already satisfied: word2number in c:\users\srini\anaconda3\lib
\site-packages (1.1)
Requirement already satisfied: contractions in c:\users\srini\anaconda3\lib
\site-packages (0.1.68)
Requirement already satisfied: textsearch>=0.0.21 in c:\users\srini\anacond
a3\lib\site-packages (from contractions) (0.0.21)
Requirement already satisfied: pyahocorasick in c:\users\srini\anaconda3\li
b\site-packages (from textsearch>=0.0.21->contractions) (1.4.4)
Requirement already satisfied: anyascii in c:\users\srini\anaconda3\lib\sit
e-packages (from textsearch>=0.0.21->contractions) (0.3.1)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14435 entries, 0 to 14434
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         14435 non-null  int64
 1   airline_sentiment  14435 non-null  object
 2   lemmatized_str     14435 non-null  object
dtypes: int64(1), object(2)
memory usage: 338.4+ KB
```

In [4]:

```
tweetsdf.drop(columns = ['Unnamed: 0' ], axis = 1, inplace = True)
```

In [6]:
```
# method to tokenize documents with lemmatization
import nltk
import string
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer

def lemma_tokenizer(corpus):   # a method to lemmatize corpus
    corpus = ''.join([ch for ch in corpus if ch not in string.punctuation])
    tokens = nltk.word_tokenize(corpus)
    lemmatizer = WordNetLemmatizer()
    return [lemmatizer.lemmatize(token) for token in tokens]

nltk_stopwords = nltk.corpus.stopwords.words('english')

# construct term-document (tf) matrix with lemmatization - not using data yet
tf = CountVectorizer(tokenizer=lemma_tokenizer,  # use lemma_tokenizer
                     stop_words=nltk_stopwords,  # use customized stopwords l
                     ngram_range=(1,2))          # use unigrams and bigrams
```

In [7]:
```
# Split the data
from sklearn.model_selection import train_test_split
textdf = tweetsdf['lemmatized_str']
ydf = tweetsdf['airline_sentiment']
textdf_train, textdf_test, ydf_train, ydf_test = train_test_split(textdf, ydf
textdf_test[:5], ydf_test[:5]
```

Out[7]:
```
(3722     united fantastic job people today ua22 dublin ...
 14175    americanair kid one answer call reserv line du...
 5619     southwestairs ceo kelly draws record crowd bwi...
 7011                cmon bad jetbae rt jetblue fleet fleek
 1224     united thanks link finally arrive brussels 9 h...
 Name: lemmatized_str, dtype: object,
 3722     non-negative
 14175        negative
 5619     non-negative
 7011         negative
 1224         negative
 Name: airline_sentiment, dtype: object)
```

In [8]:
```
tf_train = tf.fit_transform(textdf_train)
tf_train
```

```
C:\Users\srini\anaconda3\lib\site-packages\sklearn\feature_extraction\text.
py:383: UserWarning: Your stop_words may be inconsistent with your preproce
ssing. Tokenizing the stop words generated tokens ['arent', 'couldnt', 'did
nt', 'doe', 'doesnt', 'dont', 'ha', 'hadnt', 'hasnt', 'havent', 'isnt', 'mi
ghtnt', 'mustnt', 'neednt', 'shant', 'shes', 'shouldnt', 'shouldve', 'thatl
l', 'wa', 'wasnt', 'werent', 'wont', 'wouldnt', 'youd', 'youll', 'youre',
'youve'] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
```

Out[8]:
```
<11548x80867 sparse matrix of type '<class 'numpy.int64'>'
        with 215965 stored elements in Compressed Sparse Row format>
```

In [9]:
```
tf_test = tf.transform(textdf_test)
tf_test
```

Out[9]:
```
<2887x80867 sparse matrix of type '<class 'numpy.int64'>'
        with 37950 stored elements in Compressed Sparse Row format>
```

In [10]:
```
# Sentiment analysis using LSTM with supervised word embeddings

# encode label (sentiment) into a numpy arry with 0/1 values (in alphabetical
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
label_encoder.fit(ydf)
y = label_encoder.transform(ydf)
print('Encoded class order:', label_encoder.classes_)
print('Before encoding:', label_encoder.inverse_transform(y)[0:5])
print('After encoding: ', y[0:5])     # 0 = +, 1 = -

# encode Sentiment into a dataframe with 0/1 values
# y = ydf.apply(lambda x: 1 if x=='+' else 0)  # 0 = -, 1 = +
# y.head()
```

```
Encoded class order: ['negative' 'non-negative']
Before encoding: ['non-negative' 'non-negative' 'non-negative' 'negative'
'negative']
After encoding:  [1 1 1 0 0]
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(textdf, y, test_size=0.2,
X_test[:5], y_test[:5]
```

Out[11]:
```
(3722     united fantastic job people today ua22 dublin ...
 14175    americanair kid one answer call reserv line du...
 5619     southwestairs ceo kelly draws record crowd bwi...
 7011                cmon bad jetbae rt jetblue fleet fleek
 1224     united thanks link finally arrive brussels 9 h...
 Name: lemmatized_str, dtype: object,
 array([1, 0, 1, 0, 0]))
```

In [12]:
```
# tokenize documents directly using tf.keras.preprocessing.text.Tokenizer
from tensorflow.keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer()   # tokenizer = Tokenizer(num_words=100)  # to use th
tokenizer.fit_on_texts(X_train)  # tokenize the documents and index the words
```

In [13]:
```
# transform words in documents to sequences of indexes, required by tf.keras.
print('X_train in text:\n', X_train[:5])
X_train = tokenizer.texts_to_sequences(X_train)
print('\nX_train after indexing:\n', X_train[:5])
```

```
X_train in text:
 2182     united flight cancel flightled 4 hr airport bo...
13688    americanair usairways food 4 hour flight 528 c...
11312    usairways well worthless regular service bunch...
8714     jetblue supervisor humiliate u uncompromising ...
9146     usairways really one book dividend mile travel...
Name: lemmatized_str, dtype: object

X_train after indexing:
 [[2, 1, 10, 41, 85, 100, 61, 42, 90, 31, 3212, 7, 120, 3213, 1, 3214, 144
5, 77], [4, 3, 442, 85, 8, 1, 4673, 1026, 709, 2547, 1158, 922, 16, 16],
[3, 95, 1885, 3215, 11, 1446, 153, 2548, 222], [6, 511, 2549, 19, 4674, 58
1, 4675], [3, 77, 31, 42, 643, 136, 74, 690, 24, 133, 3216, 2143, 3217]]
```

In [14]:
```
# tf.keras.layers.Embedding requires inputs to have equal length; so find max
max_len = np.max([len(doc) for doc in X_train])
max_len
```

Out[14]: 24

In [15]:
```
# pad shorter documents with zeros so that all documents have max_len
from tensorflow.keras.preprocessing.sequence import pad_sequences
X_train = pad_sequences(X_train, maxlen=max_len)  # padding='pre' (default) c
print('X_train after padding:\n', X_train)
```

```
X_train after padding:
 [[   0    0    0 ... 3214 1445   77]
 [   0    0    0 ...  922   16   16]
 [   0    0    0 ...  153 2548  222]
 ...
 [   0    0    0 ...   18  274  446]
 [   0    0    0 ... 2984  107   79]
 [   0    0    0 ...  182   14  125]]
```

In [16]:
```
print('X_test in text:\n', X_test[:5])
X_test = tokenizer.texts_to_sequences(X_test)
print('\nX_test after indexing:\n', X_test[:5])
```

```
X_test in text:
 3722     united fantastic job people today ua22 dublin ...
14175    americanair kid one answer call reserv line du...
5619     southwestairs ceo kelly draws record crowd bwi...
7011                cmon bad jetbae rt jetblue fleet fleek
1224     united thanks link finally arrive brussels 9 h...
Name: lemmatized_str, dtype: object

X_test after indexing:
 [[2, 1033, 286, 88, 47, 7099, 3322, 1620, 23, 104, 718], [4, 272, 31, 129,
17, 2330, 119, 101, 466, 17, 730, 78, 178, 22, 4101], [4319, 498, 2366, 835
6, 489, 2136, 3473, 204, 1865, 1560], [1247, 50, 4499, 266, 6, 151, 152],
[2, 9, 321, 183, 150, 551, 2495, 245]]
```

In [17]:
```
X_test = pad_sequences(X_test, maxlen=max_len)
print('X_test after padding:\n', X_test)
```

```
X_test after padding:
 [[   0    0    0 ...   23  104  718]
 [   0    0    0 ...  178   22 4101]
 [   0    0    0 ...  204 1865 1560]
 ...
 [   0    0    0 ...  620   34  390]
 [   0    0    0 ...  282    1    6]
 [   0    0    0 ...  486 1138 1494]]
```

In [18]:
```
vocab_size = len(tokenizer.word_index) + 1
vocab_size
```

Out[18]: 11874

In [17]:
```
# design LSTM model with Embedding layer (which must be the 1st layer)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SpatialDropout1D, Dropout
from tensorflow.keras.layers import LSTM, Dense
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

vec_size = 10000   # dimensionality of the word vectors

model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=vec_size, input_length=m
model.add(layers.LSTM(32, return_sequences=False))
model.add(Dense(1))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 24, 10000)         118740000

 lstm (LSTM)                 (None, 32)                1284224

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 120,024,257
Trainable params: 120,024,257
Non-trainable params: 0
_________________________________________________________________
```

In [18]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accurac
model.optimizer.get_config()
```

Out[18]:
```
{'name': 'Adam',
 'learning_rate': 0.001,
 'decay': 0.0,
 'beta_1': 0.9,
 'beta_2': 0.999,
 'epsilon': 1e-07,
 'amsgrad': False}
```

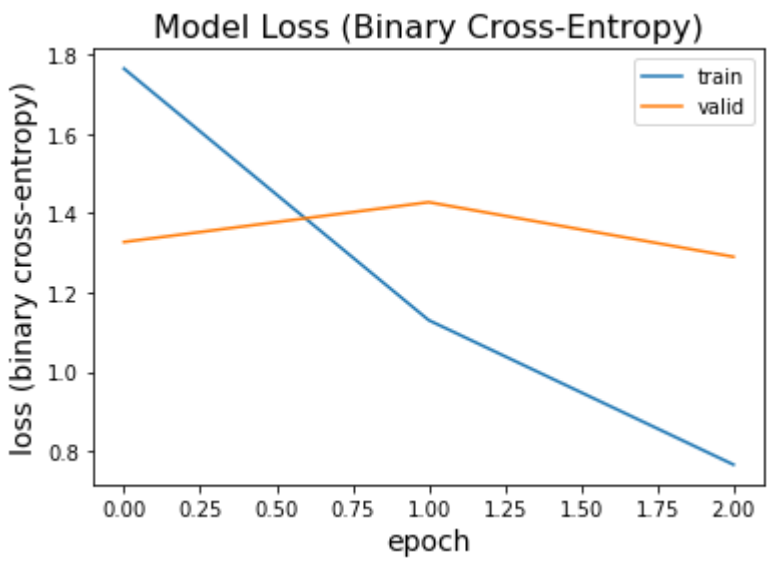
In [19]:

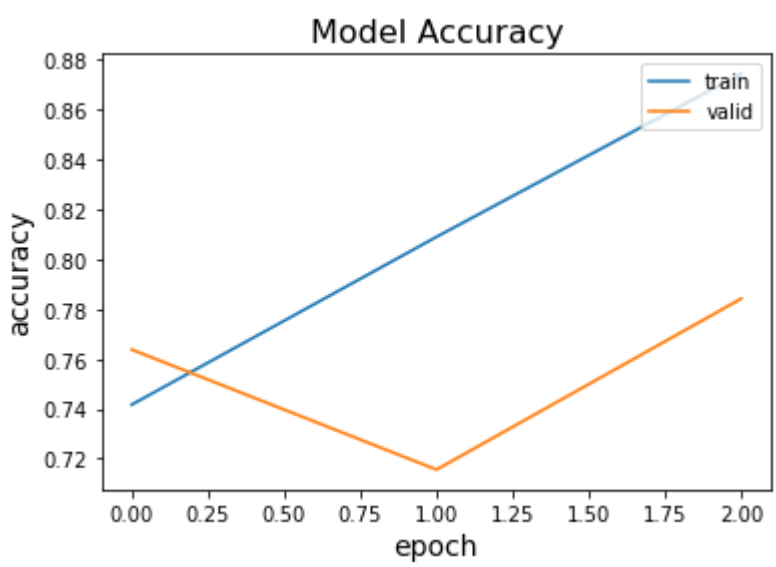
```
# train LSTM model
from time import time
t0 = time()
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), batch
print("\nTime to train LSTM model: %0.3f seconds." % (time() - t0))
```

```
Epoch 1/3
361/361 [==============================] - 800s 2s/step - loss: 1.7637 - ac
curacy: 0.7417 - val_loss: 1.3271 - val_accuracy: 0.7638
Epoch 2/3
361/361 [==============================] - 756s 2s/step - loss: 1.1301 - ac
curacy: 0.8090 - val_loss: 1.4273 - val_accuracy: 0.7156
Epoch 3/3
361/361 [==============================] - 915s 3s/step - loss: 0.7663 - ac
curacy: 0.8744 - val_loss: 1.2897 - val_accuracy: 0.7842

Time to train LSTM model: 2471.081 seconds.
```

In [20]:

```
# plot training process
%matplotlib inline
import matplotlib.pyplot as plt

# plot for loss (mse)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss (Binary Cross-Entropy)', fontsize=16)
plt.xlabel('epoch', fontsize=14)
plt.ylabel('loss (binary cross-entropy)', fontsize=14)
plt.legend(['train', 'valid'], loc='upper right')
plt.show()

# plot for metrics
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy', fontsize=16)
plt.xlabel('epoch', fontsize=14)
plt.ylabel('accuracy', fontsize=14)
plt.legend(['train', 'valid'], loc='upper right')
plt.show()
```





In [21]:

```
# Method to perform evaluation for LSTM on test data
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix

def evaluate(model, y_test, y_pred, first_label, second_label, print_predict)

    if (print_predict==True):
        print('Prediction results:')
        print('Actual:   ', np.array(y_test))
        print('Predicted:', y_pred)

    print('\nTest accuracy:', accuracy_score(y_test, y_pred))
    print('\nConfusion matrix:\n', confusion_matrix(y_test, y_pred))

    cm = pd.DataFrame(confusion_matrix(y_test, y_pred, labels=[first_label, s
                      index=['actual ' + first_label, 'actual ' + second_labe
                      columns=['predicted ' + first_label, 'predicted ' + sec
    print('\nConfusion matrix with labels:\n', cm)
    print('\nLabel ' + first_label + ' is positive class')
    TP = cm.at['actual ' + first_label,  'predicted ' + first_label]
    FP = cm.at['actual ' + second_label, 'predicted ' + first_label]
    FN = cm.at['actual ' + first_label,  'predicted ' + second_label]
    TN = cm.at['actual ' + second_label, 'predicted ' + second_label]
    print('precision =', TP/(TP+FP), ', recall =', TP/(TP+FN), ', F1-score ='
```

In [22]:

```
y_pred_prob = model.predict(X_test)
# y_pred = (y_pred_prob > 0.5).astype('int32')    # if y_pred_prob>0.5 y_pred
y_pred_prob = pd.DataFrame(y_pred_prob, columns=['y_prob'])
y_pred = y_pred_prob['y_prob'].apply(lambda x: 'negative' if x < 0.5 else 'no
y_test = label_encoder.inverse_transform(y_test)
evaluate(model, y_test, np.array(y_pred), 'negative', 'non-negative', True)
```

```
Prediction results:
Actual:    ['non-negative' 'negative' 'non-negative' ... 'negative' 'negati
ve'
 'non-negative']
Predicted: ['non-negative' 'negative' 'non-negative' ... 'negative' 'non-ne
gative'
 'non-negative']

Test accuracy: 0.7842050571527537

Confusion matrix:
 [[1592  254]
 [ 369  672]]

Confusion matrix with labels:
                     predicted negative  predicted non-negative
actual negative                    1592                     254
actual non-negative                 369                     672

Label negative is positive class
precision = 0.8118306986231515 , recall = 0.8624052004333694 , F1-score =
0.8363540845810349 Specificity = 0.6455331412103746
```

LSTM with epochs = 5 & Vec size 10000

In [2]:

```
# Package imports
import pandas as pd
import numpy as np
from time import time
from bs4 import BeautifulSoup
import spacy
!pip install unidecode
!pip install word2number
import unidecode
from word2number import w2n
!pip install contractions
import contractions

# Loading the data into pandas dataframe
tweetsdf = pd.read_csv(r"C:\Users\srini\OneDrive\Desktop\Advanced Machine Lea
tweetsdf.info()
```

```
Requirement already satisfied: unidecode in c:\users\srini\anaconda3\lib\si
te-packages (1.3.4)
Requirement already satisfied: word2number in c:\users\srini\anaconda3\lib
\site-packages (1.1)
Requirement already satisfied: contractions in c:\users\srini\anaconda3\lib
\site-packages (0.1.68)
Requirement already satisfied: textsearch>=0.0.21 in c:\users\srini\anacond
a3\lib\site-packages (from contractions) (0.0.21)
Requirement already satisfied: pyahocorasick in c:\users\srini\anaconda3\li
b\site-packages (from textsearch>=0.0.21->contractions) (1.4.4)
Requirement already satisfied: anyascii in c:\users\srini\anaconda3\lib\sit
e-packages (from textsearch>=0.0.21->contractions) (0.3.1)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14435 entries, 0 to 14434
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         14435 non-null  int64
 1   airline_sentiment  14435 non-null  object
 2   lemmatized_str     14435 non-null  object
dtypes: int64(1), object(2)
memory usage: 338.4+ KB
```

In [3]:

```
tweetsdf.drop(columns = ['Unnamed: 0' ], axis = 1, inplace = True)
```

In [5]:
```
# method to tokenize documents with lemmatization
import nltk
import string
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer

def lemma_tokenizer(corpus):   # a method to lemmatize corpus
    corpus = ''.join([ch for ch in corpus if ch not in string.punctuation])
    tokens = nltk.word_tokenize(corpus)
    lemmatizer = WordNetLemmatizer()
    return [lemmatizer.lemmatize(token) for token in tokens]

nltk_stopwords = nltk.corpus.stopwords.words('english')

# construct term-document (tf) matrix with lemmatization - not using data yet
tf = CountVectorizer(tokenizer=lemma_tokenizer,  # use lemma_tokenizer
                     stop_words=nltk_stopwords,  # use customized stopwords l
                     ngram_range=(1,2))          # use unigrams and bigrams
```

In [6]:
```
# Split the data
from sklearn.model_selection import train_test_split
textdf = tweetsdf['lemmatized_str']
ydf = tweetsdf['airline_sentiment']
textdf_train, textdf_test, ydf_train, ydf_test = train_test_split(textdf, ydf
textdf_test[:5], ydf_test[:5]
```

Out[6]:
```
(3722     united fantastic job people today ua22 dublin ...
 14175    americanair kid one answer call reserv line du...
 5619     southwestairs ceo kelly draws record crowd bwi...
 7011                cmon bad jetbae rt jetblue fleet fleek
 1224     united thanks link finally arrive brussels 9 h...
 Name: lemmatized_str, dtype: object,
 3722     non-negative
 14175        negative
 5619     non-negative
 7011         negative
 1224         negative
 Name: airline_sentiment, dtype: object)
```

In [7]:
```
tf_train = tf.fit_transform(textdf_train)
tf_train
```

```
C:\Users\srini\anaconda3\lib\site-packages\sklearn\feature_extraction\text.
py:383: UserWarning: Your stop_words may be inconsistent with your preproce
ssing. Tokenizing the stop words generated tokens ['arent', 'couldnt', 'did
nt', 'doe', 'doesnt', 'dont', 'ha', 'hadnt', 'hasnt', 'havent', 'isnt', 'mi
ghtnt', 'mustnt', 'neednt', 'shant', 'shes', 'shouldnt', 'shouldve', 'thatl
l', 'wa', 'wasnt', 'werent', 'wont', 'wouldnt', 'youd', 'youll', 'youre',
'youve'] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
```

Out[7]:
```
<11548x80867 sparse matrix of type '<class 'numpy.int64'>'
        with 215965 stored elements in Compressed Sparse Row format>
```

In [8]:
```
tf_test = tf.transform(textdf_test)
tf_test
```

Out[8]:
```
<2887x80867 sparse matrix of type '<class 'numpy.int64'>'
        with 37950 stored elements in Compressed Sparse Row format>
```

In [9]:
```
# Sentiment analysis using LSTM with supervised word embeddings

# encode label (sentiment) into a numpy arry with 0/1 values (in alphabetical
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
label_encoder.fit(ydf)
y = label_encoder.transform(ydf)
print('Encoded class order:', label_encoder.classes_)
print('Before encoding:', label_encoder.inverse_transform(y)[0:5])
print('After encoding: ', y[0:5])     # 0 = +, 1 = -

# encode Sentiment into a dataframe with 0/1 values
# y = ydf.apply(lambda x: 1 if x=='+' else 0)  # 0 = -, 1 = +
# y.head()
```

```
Encoded class order: ['negative' 'non-negative']
Before encoding: ['non-negative' 'non-negative' 'non-negative' 'negative'
'negative']
After encoding:  [1 1 1 0 0]
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(textdf, y, test_size=0.2,
X_test[:5], y_test[:5]
```

Out[10]:
```
(3722     united fantastic job people today ua22 dublin ...
 14175    americanair kid one answer call reserv line du...
 5619     southwestairs ceo kelly draws record crowd bwi...
 7011                cmon bad jetbae rt jetblue fleet fleek
 1224     united thanks link finally arrive brussels 9 h...
 Name: lemmatized_str, dtype: object,
 array([1, 0, 1, 0, 0]))
```

In [11]:
```
# tokenize documents directly using tf.keras.preprocessing.text.Tokenizer
from tensorflow.keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer()   # tokenizer = Tokenizer(num_words=100)  # to use th
tokenizer.fit_on_texts(X_train)  # tokenize the documents and index the words
```

In [12]:
```
# transform words in documents to sequences of indexes, required by tf.keras.
print('X_train in text:\n', X_train[:5])
X_train = tokenizer.texts_to_sequences(X_train)
print('\nX_train after indexing:\n', X_train[:5])
```

```
X_train in text:
 2182     united flight cancel flightled 4 hr airport bo...
13688    americanair usairways food 4 hour flight 528 c...
11312    usairways well worthless regular service bunch...
8714     jetblue supervisor humiliate u uncompromising ...
9146     usairways really one book dividend mile travel...
Name: lemmatized_str, dtype: object

X_train after indexing:
 [[2, 1, 10, 41, 85, 100, 61, 42, 90, 31, 3212, 7, 120, 3213, 1, 3214, 144
5, 77], [4, 3, 442, 85, 8, 1, 4673, 1026, 709, 2547, 1158, 922, 16, 16],
[3, 95, 1885, 3215, 11, 1446, 153, 2548, 222], [6, 511, 2549, 19, 4674, 58
1, 4675], [3, 77, 31, 42, 643, 136, 74, 690, 24, 133, 3216, 2143, 3217]]
```

In [13]:
```
# tf.keras.layers.Embedding requires inputs to have equal length; so find max
max_len = np.max([len(doc) for doc in X_train])
max_len
```

Out[13]: 24

In [14]:
```
# pad shorter documents with zeros so that all documents have max_len
from tensorflow.keras.preprocessing.sequence import pad_sequences
X_train = pad_sequences(X_train, maxlen=max_len)  # padding='pre' (default) c
print('X_train after padding:\n', X_train)
```

```
X_train after padding:
 [[   0    0    0 ... 3214 1445   77]
 [   0    0    0 ...  922   16   16]
 [   0    0    0 ...  153 2548  222]
 ...
 [   0    0    0 ...   18  274  446]
 [   0    0    0 ... 2984  107   79]
 [   0    0    0 ...  182   14  125]]
```

In [15]:
```
print('X_test in text:\n', X_test[:5])
X_test = tokenizer.texts_to_sequences(X_test)
print('\nX_test after indexing:\n', X_test[:5])
```

```
X_test in text:
 3722     united fantastic job people today ua22 dublin ...
14175    americanair kid one answer call reserv line du...
5619     southwestairs ceo kelly draws record crowd bwi...
7011                cmon bad jetbae rt jetblue fleet fleek
1224     united thanks link finally arrive brussels 9 h...
Name: lemmatized_str, dtype: object

X_test after indexing:
 [[2, 1033, 286, 88, 47, 7099, 3322, 1620, 23, 104, 718], [4, 272, 31, 129,
17, 2330, 119, 101, 466, 17, 730, 78, 178, 22, 4101], [4319, 498, 2366, 835
6, 489, 2136, 3473, 204, 1865, 1560], [1247, 50, 4499, 266, 6, 151, 152],
[2, 9, 321, 183, 150, 551, 2495, 245]]
```

In [16]:
```
X_test = pad_sequences(X_test, maxlen=max_len)
print('X_test after padding:\n', X_test)
```

```
X_test after padding:
 [[   0    0    0 ...   23  104  718]
 [   0    0    0 ...  178   22 4101]
 [   0    0    0 ...  204 1865 1560]
 ...
 [   0    0    0 ...  620   34  390]
 [   0    0    0 ...  282    1    6]
 [   0    0    0 ...  486 1138 1494]]
```

In [17]:
```
vocab_size = len(tokenizer.word_index) + 1
vocab_size
```

Out[17]:
```
11874
```

In [17]:
```
# design LSTM model with Embedding layer (which must be the 1st layer)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SpatialDropout1D, Dropout
from tensorflow.keras.layers import LSTM, Dense
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

vec_size = 10000   # dimensionality of the word vectors

model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=vec_size, input_length=m
model.add(layers.LSTM(32, return_sequences=False))
model.add(Dense(1))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 24, 10000)         118740000

 lstm (LSTM)                 (None, 32)                1284224

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 120,024,257
Trainable params: 120,024,257
Non-trainable params: 0
_________________________________________________________________
```

In [18]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accurac
model.optimizer.get_config()
```

Out[18]:
```
{'name': 'Adam',
 'learning_rate': 0.001,
 'decay': 0.0,
 'beta_1': 0.9,
 'beta_2': 0.999,
 'epsilon': 1e-07,
 'amsgrad': False}
```

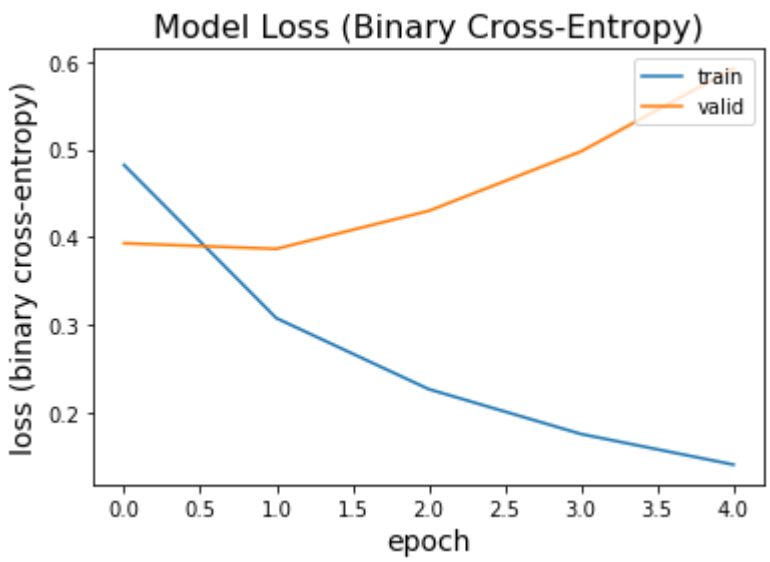
In [19]:

```
# train LSTM model
from time import time
t0 = time()
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), batch
print("\nTime to train LSTM model: %0.3f seconds." % (time() - t0))
```

```
Epoch 1/5
361/361 [==============================] - 596s 2s/step - loss: 1.4958 - ac
curacy: 0.7290 - val_loss: 1.4173 - val_accuracy: 0.7354
Epoch 2/5
361/361 [==============================] - 790s 2s/step - loss: 0.7355 - ac
curacy: 0.8427 - val_loss: 1.0415 - val_accuracy: 0.7388
Epoch 3/5
361/361 [==============================] - 745s 2s/step - loss: 0.7491 - ac
curacy: 0.8614 - val_loss: 1.4867 - val_accuracy: 0.7589
Epoch 4/5
361/361 [==============================] - 918s 3s/step - loss: 0.7950 - ac
curacy: 0.8761 - val_loss: 1.9670 - val_accuracy: 0.7371
Epoch 5/5
361/361 [==============================] - 707s 2s/step - loss: 0.8414 - ac
curacy: 0.8788 - val_loss: 1.8294 - val_accuracy: 0.7586

Time to train LSTM model: 3757.128 seconds.
```

In [20]:

```
# plot training process
%matplotlib inline
import matplotlib.pyplot as plt

# plot for loss (mse)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss (Binary Cross-Entropy)', fontsize=16)
plt.xlabel('epoch', fontsize=14)
plt.ylabel('loss (binary cross-entropy)', fontsize=14)
plt.legend(['train', 'valid'], loc='upper right')
plt.show()

# plot for metrics
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy', fontsize=16)
plt.xlabel('epoch', fontsize=14)
plt.ylabel('accuracy', fontsize=14)
plt.legend(['train', 'valid'], loc='upper right')
plt.show()
```



In [21]:

```
# Method to perform evaluation for LSTM on test data
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix

def evaluate(model, y_test, y_pred, first_label, second_label, print_predict)

    if (print_predict==True):
        print('Prediction results:')
        print('Actual:   ', np.array(y_test))
        print('Predicted:', y_pred)

    print('\nTest accuracy:', accuracy_score(y_test, y_pred))
    print('\nConfusion matrix:\n', confusion_matrix(y_test, y_pred))

    cm = pd.DataFrame(confusion_matrix(y_test, y_pred, labels=[first_label, s
                      index=['actual ' + first_label, 'actual ' + second_labe
                      columns=['predicted ' + first_label, 'predicted ' + sec
    print('\nConfusion matrix with labels:\n', cm)
    print('\nLabel ' + first_label + ' is positive class')
    TP = cm.at['actual ' + first_label,  'predicted ' + first_label]
    FP = cm.at['actual ' + second_label, 'predicted ' + first_label]
    FN = cm.at['actual ' + first_label,  'predicted ' + second_label]
    TN = cm.at['actual ' + second_label, 'predicted ' + second_label]
    print('precision =', TP/(TP+FP), ', recall =', TP/(TP+FN), ', F1-score ='
```

In [22]:

```
y_pred_prob = model.predict(X_test)
# y_pred = (y_pred_prob > 0.5).astype('int32')    # if y_pred_prob>0.5 y_pred
y_pred_prob = pd.DataFrame(y_pred_prob, columns=['y_prob'])
y_pred = y_pred_prob['y_prob'].apply(lambda x: 'negative' if x < 0.5 else 'no
y_test = label_encoder.inverse_transform(y_test)
evaluate(model, y_test, np.array(y_pred), 'negative', 'non-negative', True)
```

```
Prediction results:
Actual:    ['non-negative' 'negative' 'non-negative' ... 'negative' 'negati
ve'
 'non-negative']
Predicted: ['non-negative' 'negative' 'non-negative' ... 'negative' 'negati
ve'
 'non-negative']

Test accuracy: 0.7980602701766539

Confusion matrix:
 [[1529  317]
 [ 266  775]]

Confusion matrix with labels:
                     predicted negative  predicted non-negative
actual negative                    1529                     317
actual non-negative                 266                     775

Label negative is positive class
precision = 0.8518105849582173 , recall = 0.8282773564463706 , F1-score =
0.8398791540785498 Specificity = 0.74447646493756
```

VADER Lexicon

In [1]:
```
# Package imports
import pandas as pd
import numpy as np
from time import time
from bs4 import BeautifulSoup
import spacy
!pip install unidecode
!pip install word2number
import unidecode
from word2number import w2n
!pip install contractions
import contractions

# Loading the data into pandas dataframe
textdf = pd.read_csv(r"C:\Users\srini\OneDrive\Desktop\Advanced Machine Learn
textdf.info()
```

```
Requirement already satisfied: unidecode in c:\users\srini\anaconda3\lib\si
te-packages (1.3.4)
Requirement already satisfied: word2number in c:\users\srini\anaconda3\lib
\site-packages (1.1)
Requirement already satisfied: contractions in c:\users\srini\anaconda3\lib
\site-packages (0.1.68)
Requirement already satisfied: textsearch>=0.0.21 in c:\users\srini\anacond
a3\lib\site-packages (from contractions) (0.0.21)
Requirement already satisfied: anyascii in c:\users\srini\anaconda3\lib\sit
e-packages (from textsearch>=0.0.21->contractions) (0.3.1)
Requirement already satisfied: pyahocorasick in c:\users\srini\anaconda3\li
b\site-packages (from textsearch>=0.0.21->contractions) (1.4.4)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14435 entries, 0 to 14434
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         14435 non-null  int64
 1   airline_sentiment  14435 non-null  object
 2   lemmatized_str     14435 non-null  object
dtypes: int64(1), object(2)
memory usage: 338.4+ KB
```

In [2]:
```
textdf.drop(columns = ['Unnamed: 0' ], axis = 1, inplace = True)
```

In [5]:

```
# Sentiment analysis using VADER
from nltk.sentiment.vader import SentimentIntensityAnalyzer
def predict_sentiment(review, printout, sentiment):
    analyzer = SentimentIntensityAnalyzer()
    scores = analyzer.polarity_scores(review)  # scores has 4 values: neg, ne
    pred = 'non-negative' if scores['compound'] > 0 else 'negative'
    if (printout==True):
        print(review)
        print('neg',scores['neg'],', neu',scores['neu'],', pos',scores['pos']
        print('Predicted:', pred, ', Actual:', sentiment, '\n')
    return pred
# show results for individual reviews
for i in range(0, len(textdf[:10])):
    predict_sentiment(textdf.lemmatized_str[i], True, textdf.airline_sentimen
```

```
virginamerica dhepburn say
neg 0.0 , neu 1.0 , pos 0.0 , normalized_sum 0.0
Predicted: negative , Actual: non-negative

virginamerica plus added commercial experience tacky
neg 0.0 , neu 1.0 , pos 0.0 , normalized_sum 0.0
Predicted: negative , Actual: non-negative

virginamerica today must mean need take another trip
neg 0.0 , neu 1.0 , pos 0.0 , normalized_sum 0.0
Predicted: negative , Actual: non-negative

virginamerica really aggressive blast obnoxious entertainment guest face
little recourse
neg 0.345 , neu 0.468 , pos 0.187 , normalized_sum -0.3306
Predicted: negative , Actual: negative

virginamerica really big bad thing
neg 0.486 , neu 0.514 , pos 0.0 , normalized_sum -0.5829
Predicted: negative , Actual: negative

virginamerica seriously would pay 30 flight seat play really bad thing f
ly va
neg 0.377 , neu 0.492 , pos 0.131 , normalized_sum -0.5413
Predicted: negative , Actual: negative

virginamerica yes nearly every time fly vx ear worm go away
neg 0.0 , neu 0.787 , pos 0.213 , normalized_sum 0.4019
Predicted: non-negative , Actual: non-negative

virginamerica really miss prime opportunity men without hat parody
neg 0.158 , neu 0.585 , pos 0.256 , normalized_sum 0.2893
Predicted: non-negative , Actual: non-negative

virginamerica well notbut
neg 0.0 , neu 0.488 , pos 0.512 , normalized_sum 0.2732
Predicted: non-negative , Actual: non-negative

virginamerica amaze arrive hour early good
neg 0.0 , neu 0.385 , pos 0.615 , normalized_sum 0.7506
```

```
Predicted: non-negative , Actual: non-negative
```

In [6]:

```
# Evaluation of VADER results
text_all = textdf.lemmatized_str
y = textdf.airline_sentiment
pred_list = [predict_sentiment(review, printout=False, sentiment=y[0]) for re
predicted = pd.DataFrame(pred_list, columns=['pred'])

from sklearn import metrics
from sklearn.metrics import confusion_matrix
print('Evaluation results:\n' + 'Accuracy:', metrics.accuracy_score(y, predic
cm = pd.DataFrame(confusion_matrix(y, predicted, labels=['non-negative', 'neg
                  index=['actual non-negative', 'actual negative'],
                  columns=['predicted non-negative', 'predicted negative'])
print('Confusion matrix:\n', cm)
TP = cm.at['actual non-negative', 'predicted non-negative']
FP = cm.at['actual negative', 'predicted non-negative']
FN = cm.at['actual non-negative', 'predicted negative']
TN = cm.at['actual negative', 'predicted negative']
print('precision =', TP/(TP+FP), ', recall =', TP/(TP+FN), ', F1-score =', 2*
```

```
Evaluation results:
Accuracy: 0.5995150675441635
Confusion matrix:
                      predicted non-negative  predicted negative
actual non-negative                    3777                1571
actual negative                        4210                4877
precision = 0.47289345185927134 , recall = 0.706245325355273 , F1-score =
0.5664791901012374 , Specificity = 0.5367007813359744
```

TFIDF matrix with Lemmatization

In [7]:

```
# Compute term-document (tf) matrix and TFIDF matrix with lemmatization
import nltk
import string
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer

def lemma_tokenizer(corpus):   # a method to lemmatize corpus
    corpus = ''.join([ch for ch in corpus if ch not in string.punctuation])
    tokens = nltk.word_tokenize(corpus)
    lemmatizer = WordNetLemmatizer()
    return [lemmatizer.lemmatize(token) for token in tokens]

# use nltk stopwords list
nltk_stopwords = nltk.corpus.stopwords.words('english')
```

Split data for holdout test

In [8]:
```
# Split data for holdout test
from sklearn.model_selection import train_test_split
text_train, text_test, y_train, y_test = train_test_split(text_all, y, test_s

# Compute term-document (tf) matrix with lemmatization
tf = CountVectorizer(tokenizer=lemma_tokenizer,  # use lemma_tokenizer
                     stop_words=nltk_stopwords,  # use customized stopwords l
                     ngram_range=(1,2))          # use unigrams and bigrams
tf_train = tf.fit_transform(text_train)
tf_train
```

```
C:\Users\srini\anaconda3\lib\site-packages\sklearn\feature_extraction\text.
py:383: UserWarning: Your stop_words may be inconsistent with your preproce
ssing. Tokenizing the stop words generated tokens ['arent', 'couldnt', 'did
nt', 'doe', 'doesnt', 'dont', 'ha', 'hadnt', 'hasnt', 'havent', 'isnt', 'mi
ghtnt', 'mustnt', 'neednt', 'shant', 'shes', 'shouldnt', 'shouldve', 'thatl
l', 'wa', 'wasnt', 'werent', 'wont', 'wouldnt', 'youd', 'youll', 'youre',
'youve'] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
```

Out[8]:
```
<11548x80867 sparse matrix of type '<class 'numpy.int64'>'
        with 215965 stored elements in Compressed Sparse Row format>
```

In [13]:
```
print(text_test, '\n', y_test[:5])
tf_test = tf.transform(text_test)
tf_test[:5]
```

```
3722     united fantastic job people today ua22 dublin ...
14175    americanair kid one answer call reserv line du...
5619     southwestairs ceo kelly draws record crowd bwi...
7011                cmon bad jetbae rt jetblue fleet fleek
1224     united thanks link finally arrive brussels 9 h...
                                ...
12343    americanair 6am flight get rest wait another 2...
12462    americanair add insult injury go pick real cla...
11755        usairways miss funeral take train back newark
7389     jetblue tell guy help u sit next 5 8 year old ...
5875     southwestair ifthe80sneverstopped would still ...
Name: lemmatized_str, Length: 2887, dtype: object
 3722     non-negative
14175        negative
5619     non-negative
7011         negative
1224         negative
Name: airline_sentiment, dtype: object
```

Out[13]:
```
<5x80867 sparse matrix of type '<class 'numpy.int64'>'
        with 76 stored elements in Compressed Sparse Row format>
```

In [14]:
```
# term-document (tf) matrix for the entire corpus - for cv test
tf_all = tf.fit_transform(text_all)
tf_all
```

Out[14]:
```
<14435x97096 sparse matrix of type '<class 'numpy.int64'>'
        with 270543 stored elements in Compressed Sparse Row format>
```

In [15]:
```
# Compute tfidf matrix with lemmatization
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(tokenizer=lemma_tokenizer,  # use lemma_tokenizer
                        stop_words=nltk_stopwords,  # use customized stopwora
                        ngram_range=(1,2))          # use unigrams and bigram
tfidf_train = tfidf.fit_transform(text_train)
tfidf_train
```

Out[15]:
```
<11548x80867 sparse matrix of type '<class 'numpy.float64'>'
        with 215965 stored elements in Compressed Sparse Row format>
```

In [16]:
```
tfidf_test = tfidf.transform(text_test)
tfidf_test
```

Out[16]:
```
<2887x80867 sparse matrix of type '<class 'numpy.float64'>'
        with 37950 stored elements in Compressed Sparse Row format>
```

In [17]:
```
# tfidf matrix for the entire corpus - for cv test
tfidf_all = tfidf.fit_transform(text_all)
tfidf_all
```

Out[17]:
```
<14435x97096 sparse matrix of type '<class 'numpy.float64'>'
        with 270543 stored elements in Compressed Sparse Row format>
```

In [18]:

```
# Method to perform holdout test
def holdout_test(model, X_train, y_train, X_test, y_test, title):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(title, '\nAccuracy:', metrics.accuracy_score(y_test, y_pred))
    cm = pd.DataFrame(confusion_matrix(y_test, y_pred, labels=['non-negative'
                      index=['actual non-negative', 'actual negative'],
                      columns=['predicted non-negative', 'predicted negative'
    print('Confusion matrix:\n', cm)
    TP = cm.at['actual non-negative', 'predicted non-negative']
    FP = cm.at['actual negative', 'predicted non-negative']
    FN = cm.at['actual non-negative', 'predicted negative']
    TN = cm.at['actual negative', 'predicted negative']
    print('precision =', TP/(TP+FP), ', recall =', TP/(TP+FN), ', F1-score ='

# Method to perform cross-validation test using for loop
from sklearn.model_selection import StratifiedKFold
kf = StratifiedKFold(n_splits=5, random_state=None, shuffle=False)

def cv_test(model, X_sparse, y, title):  # X_sparse is a scipy.sparse.csr_mat
    num_total_tested = 0
    num_correctly_classified = 0   # to calculate average accuracy over k tes
    cm_sum = np.zeros((2,2)) # initialize a 2x2 confusion matrix (cm) for sum
    for train_index, test_index in kf.split(X_sparse, y):
        X_train, X_test = tf_all[train_index], tf_all[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]
        # print('X_train.shape:', X_train.shape, '\ny_train.index:', y_train.
        # print('X_test.shape:', X_test.shape, '\ny_test.index:', y_test.inde
        # print('X_train:\n', X_train, '\ny_train\n', y_train)
        # print('X_test:\n', X_test, '\ny_test\n', y_test)
        model.fit(X_train, y_train)
        num_total_tested += len(y_test)   # num_total_tested = num_total_test
        num_correctly_classified += metrics.accuracy_score(y_test, model.pred
        # print(num_total_tested, num_correctly_classified)
        cm = pd.DataFrame(confusion_matrix(y_test, model.predict(X_test), lab
                          index=['actual non-negative', 'actual negative'],
                          columns=['predicted non-negative', 'predicted negat
        # print(cm)
        cm_sum += cm

    print(title, '\nAverage accuracy:', num_correctly_classified/num_total_te
    print('Confusion matrix:\n', cm_sum)
    print("'Negative' is the Target Class")
    TP = cm.at['actual non-negative', 'predicted non-negative']
    FP = cm.at['actual negative', 'predicted non-negative']
    FN = cm.at['actual non-negative', 'predicted negative']
    TN = cm.at['actual negative', 'predicted negative']
    print('precision =', TP/(TP+FP), ', recall =', TP/(TP+FN), ', F1-score ='
```

Logistic Regression

In [19]:
```
# Analysis using logistic regression
from sklearn.linear_model import LogisticRegression
logit = LogisticRegression(random_state=1)
holdout_test(logit, tf_train, y_train, tf_test, y_test, 'Logistic regression
holdout_test(logit, tfidf_train, y_train, tfidf_test, y_test, '\nLogistic reg
```

```
Logistic regression holdout test with tf matrix:
Accuracy: 0.828195358503637
Confusion matrix:
                      predicted non-negative  predicted negative
actual non-negative                     785                 256
actual negative                         240                1606
precision = 0.7658536585365854 , recall = 0.7540826128722382 , F1-score =
0.7599225556631172 , specificity = 0.8699891657638137

Logistic reg holdout test with tfidf matrix:
Accuracy: 0.8146865258053343
Confusion matrix:
                      predicted non-negative  predicted negative
actual non-negative                     675                 366
actual negative                         169                1677
precision = 0.7997630331753555 , recall = 0.6484149855907781 , F1-score =
0.7161803713527851 , specificity = 0.9084507042253521
```

Support Vector Classifier SVC

In [20]:
```
# Analysis using support vector classifier (SVC)
from sklearn.svm import SVC
svc = SVC(kernel='linear')
holdout_test(svc, tf_train, y_train, tf_test, y_test, 'SVC holdout test with
holdout_test(svc, tfidf_train, y_train, tfidf_test, y_test, '\nSVC holdout te
```

```
SVC holdout test with tf matrix:
Accuracy: 0.8112227225493592
Confusion matrix:
                      predicted non-negative  predicted negative
actual non-negative                     802                 239
actual negative                         306                1540
precision = 0.723826714801444 , recall = 0.7704130643611912 , F1-score = 0.
7463936714751047 , specificity = 0.8342361863488624

SVC holdout test with tfidf matrix:
Accuracy: 0.828195358503637
Confusion matrix:
                      predicted non-negative  predicted negative
actual non-negative                     754                 287
actual negative                         209                1637
precision = 0.7829698857736241 , recall = 0.7243035542747358 , F1-score =
0.7524950099800399 , specificity = 0.8867822318526544
```

Decision Tree

In [21]:

```
# Analysis using decision trees
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state=1)
holdout_test(tree, tf_train, y_train, tf_test, y_test, 'Decision tree holdout
holdout_test(tree, tfidf_train, y_train, tfidf_test, y_test, '\nDecision tree
```

```
Decision tree holdout test with tf matrix:
Accuracy: 0.7519916868721856
Confusion matrix:
                      predicted non-negative  predicted negative
actual non-negative                     678                 363
actual negative                         353                1493
precision = 0.6576139670223085 , recall = 0.6512968299711815 , F1-score =
0.6544401544401545 , specificity = 0.8087757313109426

Decision tree holdout test with tfidf matrix:
Accuracy: 0.7028056806373398
Confusion matrix:
                      predicted non-negative  predicted negative
actual non-negative                     730                 311
actual negative                         547                1299
precision = 0.5716523101018011 , recall = 0.7012487992315082 , F1-score =
0.6298533218291631 , specificity = 0.7036836403033586
```

Topic Modeling with LSA

Wordcloud of the tweets

In [23]:

```
!pip install WordCloud
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
wordcloud = WordCloud(
                          collocations = False,
                          width=1600, height=800,
                          background_color='white',
                          max_words=150,
                          #max_font_size=40,
                          random_state=42
                         ).generate(' '.join(textdf['lemmatized_str'])) # can
print(wordcloud)
plt.figure(figsize=(9,8))
fig = plt.figure(1)
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

```
Requirement already satisfied: WordCloud in c:\users\srini\anaconda3\lib\si
te-packages (1.8.1)
Requirement already satisfied: numpy>=1.6.1 in c:\users\srini\anaconda3\lib
\site-packages (from WordCloud) (1.22.3)
Requirement already satisfied: matplotlib in c:\users\srini\anaconda3\lib\s
ite-packages (from WordCloud) (3.3.2)
Requirement already satisfied: pillow in c:\users\srini\anaconda3\lib\site-
packages (from WordCloud) (8.0.1)
Requirement already satisfied: certifi>=2020.06.20 in c:\users\srini\anacon
da3\lib\site-packages (from matplotlib->WordCloud) (2020.6.20)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\srini\anaco
nda3\lib\site-packages (from matplotlib->WordCloud) (2.8.1)
Requirement already satisfied: cycler>=0.10 in c:\users\srini\anaconda3\lib
\site-packages (from matplotlib->WordCloud) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
c:\users\srini\anaconda3\lib\site-packages (from matplotlib->WordCloud) (2.
4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\srini\anaconda
3\lib\site-packages (from matplotlib->WordCloud) (1.3.0)
Requirement already satisfied: six>=1.5 in c:\users\srini\anaconda3\lib\sit
e-packages (from python-dateutil>=2.1->matplotlib->WordCloud) (1.15.0)
<wordcloud.wordcloud.WordCloud object at 0x0000023A056747F0>
```

```
In [24]: # Tokenization with sklearn and nltk: set to lower case, remove stop words, a
         import nltk
         import string
         from nltk.stem import WordNetLemmatizer
         from sklearn.feature_extraction.text import CountVectorizer

         def lemma_tokenizer(corpus): # a method to lemmatize corpus
             corpus = ''.join([ch for ch in corpus if ch not in string.punctuation]) #
             tokens = nltk.word_tokenize(corpus)
             lemmatizer = WordNetLemmatizer()
             return [lemmatizer.lemmatize(token) for token in tokens]

         nltk_stopwords = nltk.corpus.stopwords.words('english') # use nltk's English
         tf = CountVectorizer(tokenizer=lemma_tokenizer, stop_words=nltk_stopwords) #
         tf_sparse = tf.fit_transform(textdf.lemmatized_str)
         tf_dictionary = tf.get_feature_names()
```

```
C:\Users\srini\anaconda3\lib\site-packages\sklearn\feature_extraction\text.
py:383: UserWarning: Your stop_words may be inconsistent with your preproce
ssing. Tokenizing the stop words generated tokens ['arent', 'couldnt', 'did
nt', 'doe', 'doesnt', 'dont', 'ha', 'hadnt', 'hasnt', 'havent', 'isnt', 'mi
ghtnt', 'mustnt', 'neednt', 'shant', 'shes', 'shouldnt', 'shouldve', 'thatl
l', 'wa', 'wasnt', 'werent', 'wont', 'wouldnt', 'youd', 'youll', 'youre',
'youve'] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
```

```
In [26]: tf = CountVectorizer(tokenizer=lemma_tokenizer, stop_words=nltk_stopwords) #
         tf_sparse = tf.fit_transform(textdf.lemmatized_str)
         tf_dictionary = tf.get_feature_names()
```

In [27]:
```
tf_dense = tf_sparse.toarray() # convert sparse to dense matrix
pd.DataFrame(tf_dense, columns=tf_dictionary)
```

Out[27]:

	0	00	0011	0016	006	0162389030167	0162424965446	0162431184663	016756007087
0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	
14430	0	0	0	0	0	0	0	0	
14431	0	0	0	0	0	0	0	0	
14432	0	0	0	0	0	0	0	0	
14433	0	0	0	0	0	0	0	0	
14434	0	0	0	0	0	0	0	0	

14435 rows × 13544 columns

In [28]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(tokenizer=lemma_tokenizer, stop_words=nltk_stopwords)
tfidf_sparse = tfidf.fit_transform(textdf.lemmatized_str)
tfidf_dictionary = tfidf.get_feature_names()
tfidf_sparse
```

```
C:\Users\srini\anaconda3\lib\site-packages\sklearn\feature_extraction\text.
py:383: UserWarning: Your stop_words may be inconsistent with your preproce
ssing. Tokenizing the stop words generated tokens ['arent', 'couldnt', 'did
nt', 'doe', 'doesnt', 'dont', 'ha', 'hadnt', 'hasnt', 'havent', 'isnt', 'mi
ghtnt', 'mustnt', 'neednt', 'shant', 'shes', 'shouldnt', 'shouldve', 'thatl
l', 'wa', 'wasnt', 'werent', 'wont', 'wouldnt', 'youd', 'youll', 'youre',
'youve'] not in stop_words.
  warnings.warn('Your stop_words may be inconsistent with '
```

Out[28]:
```
<14435x13544 sparse matrix of type '<class 'numpy.float64'>'
        with 140443 stored elements in Compressed Sparse Row format>
```

In [29]:
```
# sklearn for latent semantic analysis (LSA)
from sklearn.decomposition import TruncatedSVD
lsa = TruncatedSVD(n_components=5)
lsa
```

Out[29]:
```
TruncatedSVD(n_components=5)
```

In [30]:
```
lsa_tf_topics = lsa.fit_transform(tf_sparse)
lsa_tf_topics.shape
```

Out[30]: (14435, 5)

In [31]:
```
# print top terms for each topic
def print_top_terms(model, vocabulary, n_top_terms):
    for topic_idx, topic in enumerate(model.components_):
        message = "Topic #%d: " % topic_idx
        message += " ".join([vocabulary[i]
                             for i in topic.argsort()[:-n_top_terms - 1:-1]])
        print(message)
    print()

print('LSA topics based on term-document matrix:')
print_top_terms(lsa, tf_dictionary, 20)
```

```
LSA topics based on term-document matrix:
Topic #0: flight united usairways get americanair cancel southwestair jetbl
ue hour delay time help flightled service thanks hold 2 customer u wait
Topic #1: united bag service customer thanks get fly plane time airline gat
e make would delay seat unite lose wait thank check
Topic #2: flight cancel united flightled jetblue southwestair late attendan
t flighted book virginamerica tomorrow delayed problem sfo fll reschedule c
ancelled den bna
Topic #3: usairways flight united hold late clt phl miss delay charlotte mi
nute mile min philly connection dca hour attendant connect usair
Topic #4: jetblue usairways southwestair fly delay thanks fleek fleet time
go would thank jfk plane great u love service guy know
```

In [32]:
```
# combining all the string from topics 1 - 5 for visualizing in wordcloud
Top_words_LSA_TF = "flight united usairways get americanair cancel southwesta
```

Word Cloud for Top words of LSA with TF matrix

In [34]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
wordcloud = WordCloud(
                          collocations = False,
                          width=1600, height=800,
                          background_color='white',
                          max_words=150,
                          #max_font_size=40,
                          random_state=42
                         ).generate(' '.join([Top_words_LSA_TF])) # can't pas
print(wordcloud)
plt.figure(figsize=(9,8))
fig = plt.figure(1)
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

```
<wordcloud.wordcloud.WordCloud object at 0x0000023A26A22BB0>
```

In [35]:
```
lsa.fit_transform(tfidf_sparse)
print('LSA topics based on tfidf matrix:')
print_top_terms(lsa, tfidf_dictionary, 20)
```

```
LSA topics based on tfidf matrix:
Topic #0: flight united usairways americanair get southwestair thanks jetbl
ue cancel hour help thank hold service time customer call delay flightled w
ait
Topic #1: jetblue fleek fleet thanks thank rt great united much send fly gu
y love good dm awesome follow southwestair jfk stop
Topic #2: united thank thanks southwestair dm customer service follow send
much bag great response bad appreciate airline yes fly ever make
Topic #3: united flight cancel delay flightled jetblue fleek fleet late mis
s book plane seat tomorrow unite problem bag airline gate check
Topic #4: southwestair thank flight cancel flightled dm tomorrow follow lov
e send destinationdragons rebook much virginamerica flighted southwest book
today please imaginedragons
```

In [36]:
```
# combining all the string from topics 1 - 5 for visualizing in wordcloud
Top_words_LSA_TFIDF = 'flight united usairways americanair get southwestair t
```

Word Cloud for Top words of LSA with TFIDF matrix

In [37]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
wordcloud = WordCloud(
                          collocations = False,
                          width=1600, height=800,
                          background_color='white',
                          max_words=150,
                          #max_font_size=40,
                          random_state=42
                         ).generate(' '.join([Top_words_LSA_TFIDF])) # can't
print(wordcloud)
plt.figure(figsize=(9,8))
fig = plt.figure(1)
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

```
<wordcloud.wordcloud.WordCloud object at 0x0000023A26E55D00>
```