Sentiment Analysis on US Airline Reviews

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense, Dropout, SpatialDropout1D
from tensorflow.keras.layers import Embedding

df = pd.read_csv(r"C:\Users\user\Desktop\Tweets.csv")
```

In [2]:

```
df.head()
```

Out[2]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason
0	5.703060e+17	neutral	1.0000	NaN	
1	5.703010e+17	positive	0.3486	NaN	
2	5.703010e+17	neutral	0.6837	NaN	
3	5.703010e+17	negative	1.0000	Bad Flight	
4	5.703010e+17	negative	1.0000	Can't Tell	

In [3]:

```
df.columns
```

Out[3]:

```
Index(['tweet_id', 'airline_sentiment', 'airline_sentiment_confidence',
       'negativereason', 'negativereason_confidence', 'airline',
       'airline_sentiment_gold', 'name', 'negativereason_gold',
       'retweet_count', 'text', 'tweet_coord', 'tweet_created',
       'tweet_location', 'user_timezone'],
      dtype='object')
```

In [4]:

```
tweet_df = df[['text','airline_sentiment']]
print(tweet_df.shape)
tweet_df.head(5)
```

```
(14640, 2)
```

Out[4]:

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

In [5]:

```
tweet_df = tweet_df[tweet_df['airline_sentiment'] != 'neutral']
print(tweet_df.shape)
tweet_df.head(5)
```

```
(11541, 2)
```

Out[5]:

	text	airline_sentiment
1	@VirginAmerica plus you've added commercials t...	positive
3	@VirginAmerica it's really aggressive to blast...	negative
4	@VirginAmerica and it's a really big bad thing...	negative
5	@VirginAmerica seriously would pay $30 a fligh...	negative
6	@VirginAmerica yes, nearly every time I fly VX...	positive

In [6]:

```
tweet_df["airline_sentiment"].value_counts()
```

Out[6]:

```
negative    9178
positive    2363
Name: airline_sentiment, dtype: int64
```

In [7]:

```
sentiment_label = tweet_df.airline_sentiment.factorize()
sentiment_label
```

Out[7]:

```
(array([0, 1, 1, ..., 0, 1, 1], dtype=int64),
 Index(['positive', 'negative'], dtype='object'))
```

In [8]:

```
tweet = tweet_df.text.values
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(tweet)
vocab_size = len(tokenizer.word_index) + 1
encoded_docs = tokenizer.texts_to_sequences(tweet)
padded_sequence = pad_sequences(encoded_docs, maxlen=200)
```

In [9]:

```
print(tokenizer.word_index)
```

```
{'to': 1, 'the': 2, 'i': 3, 'a': 4, 'united': 5, 'you': 6, 'for': 7, 'flig
ht': 8, 'and': 9, 'on': 10, 'my': 11, 'usairways': 12, 'americanair': 13,
'is': 14, 'in': 15, 'southwestair': 16, 'of': 17, 'jetblue': 18, 'me': 19,
'your': 20, 'it': 21, 'was': 22, 'not': 23, 'no': 24, 'have': 25, 'at': 2
6, 'with': 27, 'that': 28, 'this': 29, 'get': 30, 'but': 31, 'be': 32, 'ca
ncelled': 33, 'thanks': 34, 'now': 35, 'service': 36, 'are': 37, 'we': 38,
'from': 39, 'an': 40, 'been': 41, 'just': 42, '2': 43, 'so': 44, 'custome
r': 45, 'help': 46, 't': 47, 'can': 48, 'time': 49, 'co': 50, 'up': 51, 'h
ours': 52, 'http': 53, 'do': 54, 'hold': 55, 'they': 56, 'out': 57, 'amp':
58, 'plane': 59, "i'm": 60, 'us': 61, 'all': 62, 'will': 63, 'why': 64, 't
hank': 65, 'still': 66, 'our': 67, 'delayed': 68, 'what': 69, 'when': 70,
'how': 71, 'one': 72, "can't": 73, 'flights': 74, 'call': 75, 'gate': 76,
'hour': 77, 'had': 78, 'flightled': 79, 'back': 80, 'bag': 81, 'if': 82,
'would': 83, 'after': 84, 'has': 85, 'about': 86, 'there': 87, "it's": 88,
"don't": 89, 'as': 90, 'got': 91, 'late': 92, 'phone': 93, 'need': 94, 'pl
ease': 95, 'again': 96, '3': 97, 'airline': 98, 'over': 99, 'or': 100, 'li
ke': 101, 'waiting': 102, 'virginamerica': 103, 'today': 104, 'more': 105,
'am': 106, 'guys': 107, 'by': 108, '4': 109, 'great': 110, 'wait': 111,
'u': 112, 'because': 113, 'fly': 114, 'never': 115, 'day': 116, 'trying':
```

In [10]:

```
print(tweet[0])
print(encoded_docs[0])
```

```
@VirginAmerica plus you've added commercials to the experience... tacky.
[103, 575, 530, 1287, 2416, 1, 2, 177]
```

In [11]:

```
print(padded_sequence[0])
```

```
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0  103  575  530 1287
 2416    1    2  177]
```

In [12]:

```
embedding_vector_length = 32
model = Sequential()
model.add(Embedding(vocab_size, embedding_vector_length, input_length=200) )
model.add(SpatialDropout1D(0.25))
model.add(LSTM(50, dropout=0.5, recurrent_dropout=0.5))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 200, 32)           423488    
                                                                 
 spatial_dropout1d (SpatialD  (None, 200, 32)          0         
 ropout1D)                                                       
                                                                 
 lstm (LSTM)                 (None, 50)                16600     
                                                                 
 dropout (Dropout)           (None, 50)                0         
                                                                 
 dense (Dense)               (None, 1)                 51        
                                                                 
=================================================================
Total params: 440,139
Trainable params: 440,139
Non-trainable params: 0
_________________________________________________________________
None
```

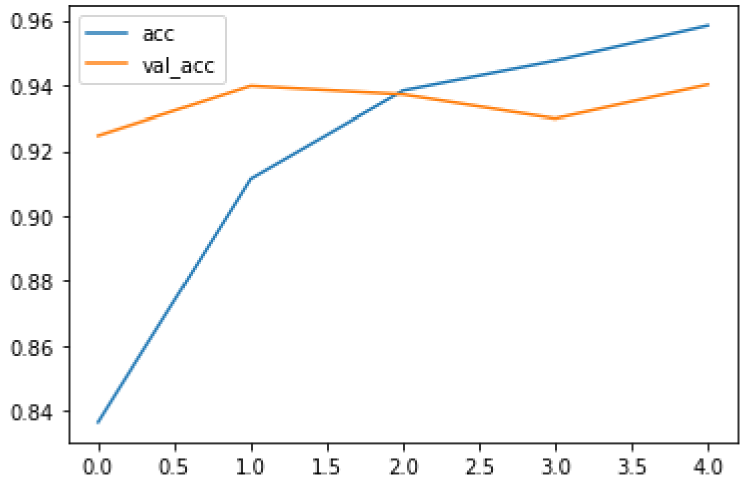
In [13]:

```
history = model.fit(padded_sequence,sentiment_label[0],validation_split=0.2, epochs=5, batc
```

```
Epoch 1/5
289/289 [==============================] - 99s 328ms/step - loss: 0.3891 - a
ccuracy: 0.8365 - val_loss: 0.2071 - val_accuracy: 0.9246
Epoch 2/5
289/289 [==============================] - 99s 341ms/step - loss: 0.2251 - a
ccuracy: 0.9114 - val_loss: 0.1667 - val_accuracy: 0.9398
Epoch 3/5
289/289 [==============================] - 98s 338ms/step - loss: 0.1622 - a
ccuracy: 0.9384 - val_loss: 0.1683 - val_accuracy: 0.9372
Epoch 4/5
289/289 [==============================] - 99s 343ms/step - loss: 0.1359 - a
ccuracy: 0.9476 - val_loss: 0.1784 - val_accuracy: 0.9298
Epoch 5/5
289/289 [==============================] - 101s 350ms/step - loss: 0.1161 -
accuracy: 0.9584 - val_loss: 0.1796 - val_accuracy: 0.9402
```

In [14]:

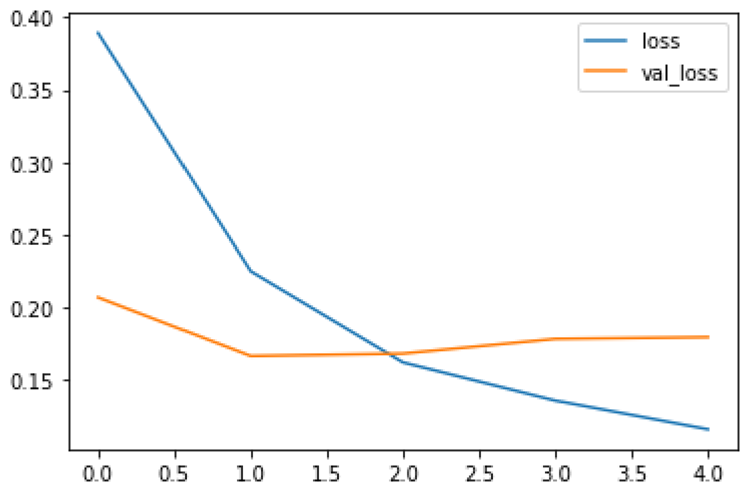
```
plt.plot(history.history['accuracy'], label='acc')
plt.plot(history.history['val_accuracy'], label='val_acc')
plt.legend()
plt.show()
plt.savefig("Accuracy plot.jpg")
```



```
<Figure size 432x288 with 0 Axes>
```

In [15]:

```
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.legend()
plt.show()
plt.savefig("Loss plot.jpg")
```



```
<Figure size 432x288 with 0 Axes>
```

In [16]:

```
def predict_sentiment(text):
    tw = tokenizer.texts_to_sequences([text])
    tw = pad_sequences(tw,maxlen=200)
    prediction = int(model.predict(tw).round().item())
    print("Predicted label: ", sentiment_label[1][prediction])
```

In [17]:

```
test_sentence1 = "I enjoyed my journey on this flight."
predict_sentiment(test_sentence1)

test_sentence2 = "This is the worst flight experience of my life!"
predict_sentiment(test_sentence2)
```

```
Predicted label:  positive
Predicted label:  negative
```

In [ ]:

```

```