In [1]:

```python

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import plotly.offline as py
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.express as px

from sklearn.feature_extraction.text import CountVectorizer
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
import re

```

In [2]:

```python
mydata = pd.read_csv('Sentiment.csv')
mydata = mydata[['text','sentiment']] #Selecting the required columns
```

In [3]:
```
mydata.head(15)
```

Out[3]:

| | text | sentiment |
|---|---|---|
| 0 | RT @NancyLeeGrahn: How did everyone feel about... | Neutral |
| 1 | RT @ScottWalker: Didn't catch the full #GOPdeb... | Positive |
| 2 | RT @TJMShow: No mention of Tamir Rice and the ... | Neutral |
| 3 | RT @RobGeorge: That Carly Fiorina is trending ... | Positive |
| 4 | RT @DanScavino: #GOPDebate w/ @realDonaldTrump... | Positive |
| 5 | RT @GregAbbott_TX: @TedCruz: "On my first day ... | Positive |
| 6 | RT @warriorwoman91: I liked her and was happy ... | Negative |
| 7 | Going on #MSNBC Live with @ThomasARoberts arou... | Neutral |
| 8 | Deer in the headlights RT @lizzwinstead: Ben C... | Negative |
| 9 | RT @NancyOsborne180: Last night's debate prove... | Negative |
| 10 | @JGreenDC @realDonaldTrump In all fairness #Bi... | Negative |
| 11 | RT @WayneDupreeShow: Just woke up to tweet thi... | Positive |
| 12 | Me reading my family's comments about how grea... | Negative |
| 13 | RT @ArcticFox2016: RT @AllenWestRepub "Dear @J... | Neutral |
| 14 | RT @pattonoswalt: I loved Scott Walker as Mark... | Positive |

#1) Print the total number of positive and negative sentiments.

After observing the above dataset, Let's drop the 'Neutral' sentiments as the objective is to only calculate positive and negative tweets. Then let's filter the tweets so that only valid texts and words remain. Finally let's define the number of max features as 2000 and use Tokenizer to vectorize and convert text into Sequences so that the Network can deal with it as input.

In [4]:

```
mydata = mydata[mydata.sentiment != "Neutral"]
mydata['text'] = mydata['text'].apply(lambda x: x.lower())
mydata['text'] = mydata['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]','',x)))

print(mydata[ mydata['sentiment'] == 'Positive'].size)
print(mydata[ mydata['sentiment'] == 'Negative'].size)

for idx,row in mydata.iterrows():
    row[0] = row[0].replace('rt',' ')

max_fatures = 2000
tokenizer = Tokenizer(num_words=max_fatures, split=' ')
tokenizer.fit_on_texts(mydata['text'].values)
X = tokenizer.texts_to_sequences(mydata['text'].values)
X = pad_sequences(X)
```

```
4472
16986
```

In [5]:

```
fig = px.histogram(mydata, x="sentiment")
fig.update_traces(marker_color="turquoise",marker_line_color='rgb(8,48,107)',
                  marker_line_width=1.5)
fig.update_layout(title_text='Analyzing Different Sentiments')
fig.show()
```

The above plot shows the count of positive and negative sentiments available in the dataset.

#2) Build a sequential LSTM model to predict positive and negative sentiments.

In [6]:
```
embed_dim = 128
lstm_out = 196

model = Sequential()
model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1]))
model.add(SpatialDropout1D(0.4))
model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(2,activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 28, 128)           256000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 28, 128)           0
_________________________________________________________________
lstm (LSTM)                  (None, 196)               254800
_________________________________________________________________
dense (Dense)                (None, 2)                 394
=================================================================
Total params: 511,194
Trainable params: 511,194
Non-trainable params: 0
_________________________________________________________________
None
```

Hereby I declare the train and test dataset.

In [8]:
```
Y = pd.get_dummies(mydata['sentiment']).values
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.3, random_state = 40)
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

```
(7510, 28) (7510, 2)
(3219, 28) (3219, 2)
```

Now, we train the model. Let's run 10 epoch

In [10]:
```
batch_size = 32
model.fit(X_train, Y_train, epochs = 10, batch_size=batch_size, verbose = 2)
```

```
Epoch 1/10
235/235 - 35s - loss: 0.4201 - accuracy: 0.8206
Epoch 2/10
235/235 - 35s - loss: 0.3169 - accuracy: 0.8643
Epoch 3/10
235/235 - 35s - loss: 0.2733 - accuracy: 0.8846
Epoch 4/10
235/235 - 35s - loss: 0.2439 - accuracy: 0.8968
Epoch 5/10
235/235 - 35s - loss: 0.2157 - accuracy: 0.9067
Epoch 6/10
235/235 - 35s - loss: 0.1913 - accuracy: 0.9210
Epoch 7/10
235/235 - 35s - loss: 0.1825 - accuracy: 0.9244
Epoch 8/10
235/235 - 35s - loss: 0.1592 - accuracy: 0.9318
Epoch 9/10
235/235 - 35s - loss: 0.1485 - accuracy: 0.9366
Epoch 10/10
235/235 - 35s - loss: 0.1329 - accuracy: 0.9422
```

Out[10]:
```
<keras.callbacks.History at 0x7fc17772c310>
```

# Validation set extraction

# Also measuring score and accuracy.

In [11]:

```
validation_size = 1500

X_validate = X_test[-validation_size:]
Y_validate = Y_test[-validation_size:]
X_test = X_test[:-validation_size]
Y_test = Y_test[:-validation_size]
score,acc = model.evaluate(X_test, Y_test, verbose = 2, batch_size = batch_size)
print("score: %.2f" % (score))
print("acc: %.2f" % (acc))
```

```
54/54 - 2s - loss: 0.5102 - accuracy: 0.8482
score: 0.51
acc: 0.85
```

Finally measuring the number of correct guesses. It is clear that finding negative tweets goes very well for the Network but deciding whether is positive is not really. My educated guess here is that the positive training set is dramatically smaller than the negative, hence the "bad" results for positive tweets.

In [12]:

```
pos_cnt, neg_cnt, pos_correct, neg_correct = 0, 0, 0, 0
for x in range(len(X_validate)):

    result = model.predict(X_validate[x].reshape(1,X_test.shape[1]),batch_size=1,verbose = 2)[0]

    if np.argmax(result) == np.argmax(Y_validate[x]):
        if np.argmax(Y_validate[x]) == 0:
            neg_correct += 1
        else:
            pos_correct += 1

    if np.argmax(Y_validate[x]) == 0:
        neg_cnt += 1
    else:
        pos_cnt += 1



print("pos_acc", pos_correct/pos_cnt*100, "%")
print("neg_acc", neg_correct/neg_cnt*100, "%")

```

```
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
1/1 - 0s
```

# 3) Based on the model, check the sentiment for the following two sentences

# a. 'He is a great leader.'

# b. 'He is a terrible leader.'

In [16]:

```
check_senti = ['He is a great leader.']
#vectorizing the tweet by the pre-fitted tokenizer instance
check_senti = tokenizer.texts_to_sequences(check_senti)
#padding the tweet to have exactly the same shape as `embedding_2` input
check_senti = pad_sequences(check_senti, maxlen=28, dtype='int32', value=0)
print(check_senti)
sentiment = model.predict(check_senti,batch_size=1,verbose = 2)[0]
if(np.argmax(sentiment) == 0):
    print("negative")
elif (np.argmax(sentiment) == 1):
    print("positive")
```

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0  32   5   7 146 340]]
1/1 - 0s
positive
```

In [17]:

```
check_senti1 = ['He is a terrible leader.']
#vectorizing the tweet by the pre-fitted tokenizer instance
check_senti1 = tokenizer.texts_to_sequences(check_senti1)
#padding the tweet to have exactly the same shape as `embedding_2` input
check_senti1 = pad_sequences(check_senti1, maxlen=28, dtype='int32', value=0)
print(check_senti1)
sentiment = model.predict(check_senti1,batch_size=1,verbose = 2)[0]
if(np.argmax(sentiment) == 0):
    print("negative")
elif (np.argmax(sentiment) == 1):
    print("positive")
```

```
[[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0   32    5    7 1003  340]]
1/1 - 0s
negative
```