Twitter Sentiment Analysis

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import csv
import re
import random
import numpy as np

from IPython import embed
```

Data Analysis and Visualization

In [2]:
```
data = pd.read_csv('Sentiment Analysis Dataset.csv',error_bad_lines=False)
```

```
b'Skipping line 8836: expected 4 fields, saw 5\n'
b'Skipping line 535882: expected 4 fields, saw 7\n'
```

In [3]:
```
print(data.shape)
```

```
(1578612, 4)
```

In [4]:
```
data.head()
```

Out[4]:

	ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140	is so sad for my APL frie...
1	2	0	Sentiment140	I missed the New Moon trail...
2	3	1	Sentiment140	omg its already 7:30 :O
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I'...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! ...

In [5]:
```
data.isnull().any()
```

Out[5]:
```
ItemID             False
Sentiment          False
SentimentSource    False
SentimentText      False
dtype: bool
```

In [6]:
```
# checking out the negative comments from the train set
data[data['Sentiment'] == 0].head(10)
```

Out[6]:

	ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140	is so sad for my APL frie...
1	2	0	Sentiment140	I missed the New Moon trail...
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I'...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! ...
5	6	0	Sentiment140	or i just worry too much?
7	8	0	Sentiment140	Sunny Again Work Tomorrow :-| ...
10	11	0	Sentiment140	I must think about positive..
12	13	0	Sentiment140	this weekend has sucked so far
13	14	0	Sentiment140	jb isnt showing in australia any more!
14	15	0	Sentiment140	ok thats it you win.

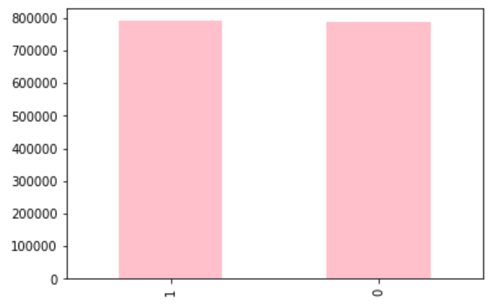
In [7]:
```
#checking out the postive comments from the train set
data[data['Sentiment'] == 1].head(10)
```

Out[7]:

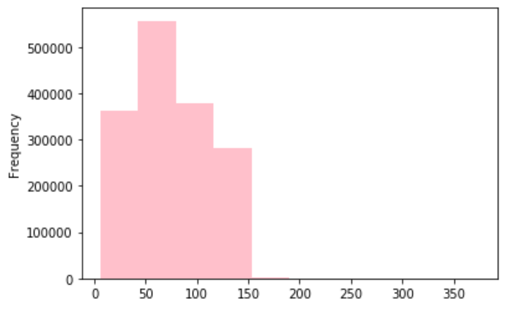
	ItemID	Sentiment	SentimentSource	SentimentText
2	3	1	Sentiment140	omg its already 7:30 :O
6	7	1	Sentiment140	Juuuuuuuuuuuuuuuuussssst Chillin!!
8	9	1	Sentiment140	handed in my uniform today . i miss you ...
9	10	1	Sentiment140	hmmmm.... i wonder how she my number @-)
11	12	1	Sentiment140	thanks to all the haters up in my face a...
17	18	1	Sentiment140	Feeling strangely fine. Now I'm gonna go l...
22	23	1	Sentiment140	You're the only one who can see this cause...
28	29	1	Sentiment140	goodbye exams, HELLO ALCOHOL TONIGHT
38	39	1	Sentiment140	uploading pictures on friendster
41	42	1	Sentiment140	(: !!!!!! - so i wrote something last week. ...

```
In [8]: data['Sentiment'].value_counts().plot.bar(color = 'pink', figsize = (6, 4))
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0xb5dd9c8>
```



```
In [9]: # checking the distribution of tweets in the data
        length_train = data['SentimentText'].str.len().plot.hist(color = 'pink', figsize = (6, 4))
```



```
In [10]: data.groupby('Sentiment').describe()
```

Out[10]:

	ItemID							
	count	mean	std	min	25%	50%	75%	max
Sentiment								
0	788435.0	840458.653997	449231.963498	1.0	474930.0	868240.0	1223861.5	1578627.0
1	790177.0	738294.923096	456402.871802	3.0	339362.0	706525.0	1135595.0	1578624.0

Data Cleaning

In [11]:
```python
def clean_str(string):


    #EMOJIS
    string = re.sub(r":\)","emojihappy1",string)
    string = re.sub(r":P","emojihappy2",string)
    string = re.sub(r":p","emojihappy3",string)
    string = re.sub(r":>","emojihappy4",string)
    string = re.sub(r":3","emojihappy5",string)
    string = re.sub(r":D","emojihappy6",string)
    string = re.sub(r" XD ","emojihappy7",string)
    string = re.sub(r" <3 ","emojihappy8",string)

    string = re.sub(r":\(","emojisad9",string)
    string = re.sub(r":<","emojisad10",string)
    string = re.sub(r":<","emojisad11",string)
    string = re.sub(r">:\(","emojisad12",string)

    #MENTIONS "(@)\w+"
    string = re.sub(r"(@)\w+","",string)

    #WEBSITES
    string = re.sub(r"http(s)*:(\S)*","linktoken",string)

    #STRANGE UNICODE \x...
    string = re.sub(r"\\x(\S)*","",string)

    #General Cleanup and Symbols
    string = re.sub(r"[^A-Za-z0-9(),!?\'\`]", " ", string)
    string = re.sub(r"\'s", " \'s", string)
    string = re.sub(r"\'ve", " \'ve", string)
    string = re.sub(r"n\'t", " n\'t", string)
    string = re.sub(r"\'re", " \'re", string)
    string = re.sub(r"\'d", " \'d", string)
    string = re.sub(r"\'ll", " \'ll", string)
    string = re.sub(r",", " , ", string)
    string = re.sub(r"!", " ! ", string)
    string = re.sub(r"\(", " \( ", string)
    string = re.sub(r"\)", " \) ", string)
    string = re.sub(r"\?", " \? ", string)
    string = re.sub(r"\s{2,}", " ", string)

    return string.strip().lower()
```

In [12]:
```python
data['clean_text'] = data['SentimentText'].apply(clean_str)
```

In [13]:
```python
data.head()
```

Out[13]:

	ItemID	Sentiment	SentimentSource	SentimentText	clean_text
0	1	0	Sentiment140	is so sad for my APL frie...	is so sad for my apl friend
1	2	0	Sentiment140	I missed the New Moon trail...	i missed the new moon trailer
2	3	1	Sentiment140	omg its already 7:30 :O	omg its already 7emojihappy50 o
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I'...	omgaga im sooo im gunna cry i 've been at this...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! ...	i think mi bf is cheating on me ! ! ! t t

In [14]:
```python
from sklearn.feature_extraction.text import CountVectorizer


cv = CountVectorizer(stop_words = 'english')
words = cv.fit_transform(data.clean_text)

sum_words = words.sum(axis=0)

words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)

frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])

frequency.head(30).plot(x='word', y='freq', kind='bar', figsize=(15, 7), color = 'blue')
plt.title("Most Frequently Occuring Words - Top 30")
```

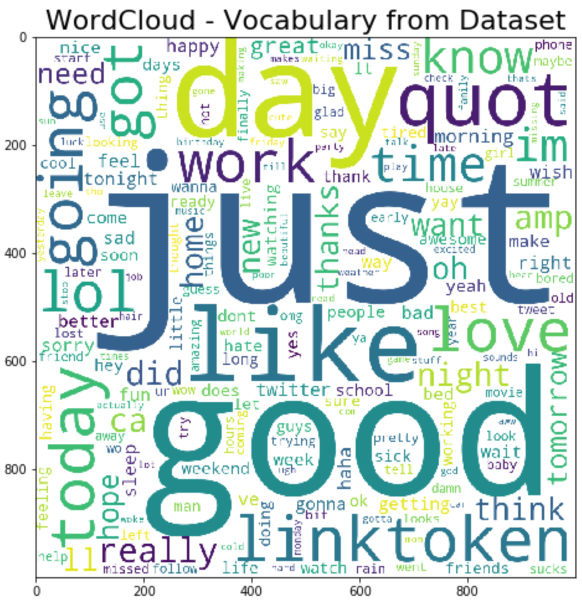
Out[14]:
```
Text(0.5, 1.0, 'Most Frequently Occuring Words - Top 30')
```

```
In [19]: pip install wordcloud
```

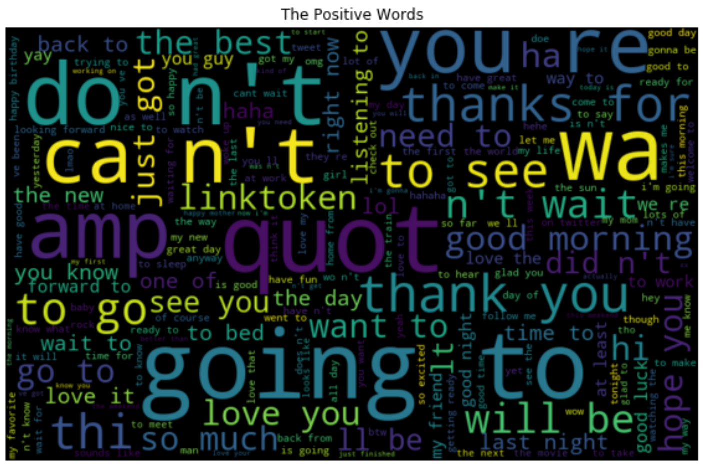
```
Requirement already satisfied: wordcloud in c:\users\srikanth\anaconda3\lib\site-packages (1.7.0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\srikanth\anaconda3\lib\site-packages (from wordcloud) (1.18.1)
Requirement already satisfied: pillow in c:\users\srikanth\anaconda3\lib\site-packages (from wordcloud) (7.0.0)
Requirement already satisfied: matplotlib in c:\users\srikanth\anaconda3\lib\site-packages (from wordcloud) (3.1.3)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\srikanth\anaconda3\lib\site-packages (from matplotlib->wordclou
d) (2.8.1)
Requirement already satisfied: cycler>=0.10 in c:\users\srikanth\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.10.
0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\srikanth\anaconda3\lib\site-packages (from m
atplotlib->wordcloud) (2.4.6)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\srikanth\anaconda3\lib\site-packages (from matplotlib->wordcloud)
(1.1.0)
Requirement already satisfied: six>=1.5 in c:\users\srikanth\anaconda3\lib\site-packages (from python-dateutil>=2.1->matplotlib-
>wordcloud) (1.14.0)
Requirement already satisfied: setuptools in c:\users\srikanth\anaconda3\lib\site-packages (from kiwisolver>=1.0.1->matplotlib->
wordcloud) (45.2.0.post20200210)
Note: you may need to restart the kernel to use updated packages.
```

```
In [17]: from wordcloud import WordCloud

         wordcloud = WordCloud(background_color = 'white', width = 1000, height = 1000).generate_from_frequencies(dict(words_freq))

         plt.figure(figsize=(10,8))
         plt.imshow(wordcloud)
         plt.title("WordCloud - Vocabulary from Dataset", fontsize = 22)
```

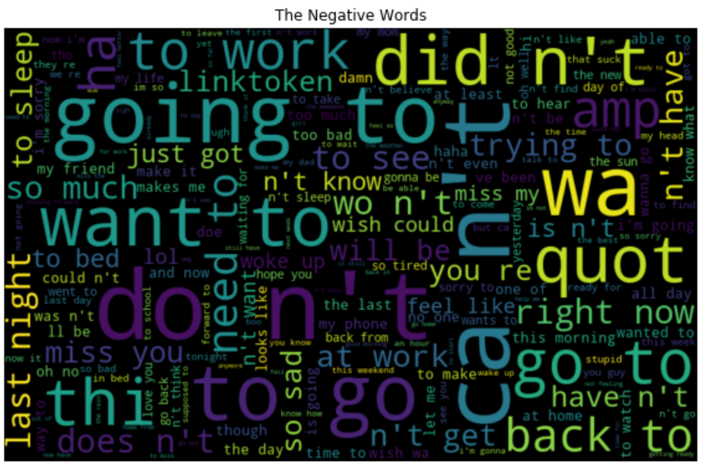
```
Out[17]: Text(0.5, 1.0, 'WordCloud - Vocabulary from Dataset')
```



```
In [18]: positive_words =' '.join([text for text in data['clean_text'][data['Sentiment'] == 1]])

         wordcloud = WordCloud(width=800, height=500, random_state = 0, max_font_size = 110).generate(positive_words)
         plt.figure(figsize=(10, 7))
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis('off')
         plt.title('The Positive Words')
         plt.show()
```



In [20]:
```python
negative_words =' '.join([text for text in data['clean_text'][data['Sentiment'] == 0]])

wordcloud = WordCloud(width=800, height=500, random_state = 0, max_font_size = 110).generate(negative_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.title('The Negative Words')
plt.show()
```



In [23]:
```python
pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\srikanth\anaconda3\lib\site-packages (3.4.5)
Requirement already satisfied: six in c:\users\srikanth\anaconda3\lib\site-packages (from nltk) (1.14.0)
Note: you may need to restart the kernel to use updated packages.
```

In [27]:
```python
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from textblob import TextBlob
```

In [28]:
```python
def form_sentence(tweet):
    tweet_blob = TextBlob(tweet)
    return ' '.join(tweet_blob.words)

print(form_sentence(data['SentimentText'].iloc[0]))
print(data['SentimentText'].iloc[0])
```

```
is so sad for my APL friend
                     is so sad for my APL friend.............
```

In [30]:
```python
def no_user_alpha(tweet):
    tweet_list = [ele for ele in tweet.split() if ele != 'user']
    clean_tokens = [ele for ele in tweet.split() if re.match(r'[^\W\d]*$', ele)]
    clean_s = ' '.join(clean_tokens)
    clean_mess = [word for word in clean_s.split() if word.lower() not in stopwords.words('english')]
    return ' '.join(clean_mess)
print(no_user_alpha(form_sentence(data['SentimentText'].iloc[0])))
print(data['SentimentText'].iloc[0])
```

```
sad APL friend
                     is so sad for my APL friend.............
```

In [31]:
```python
def normalization(tweet_list):
        lem = WordNetLemmatizer()
        normalized_tweet = []
        for word in tweet_list:
            normalized_text = lem.lemmatize(word,'v')
            normalized_tweet.append(normalized_text)
        return ' '.join(normalized_tweet)
print(normalization(data['SentimentText'].iloc[0].split()))
print(data['SentimentText'].iloc[0])
```

```
be so sad for my APL friend.............
                     is so sad for my APL friend.............
```

In [32]:
```python
data['clean_text'] = data['clean_text'].apply(form_sentence)
```

In [ ]:
```python
#data['clean_text'] = data['clean_text'].apply(no_user_alpha)
```

In [33]:
```python
data['clean_text'] = data['clean_text'].apply(normalization)
```

```
In [34]: data.head()
```

Out[34]:

	ItemID	Sentiment	SentimentSource	SentimentText	clean_text
0	1	0	Sentiment140	is so sad for my APL frie...	i s s o s a d f o r m y a p l f r ...
1	2	0	Sentiment140	I missed the New Moon trail...	i m i s s e d t h e n e w m o o n t ...
2	3	1	Sentiment140	omg its already 7:30 :O	o m g i t s a l r e a d y 7 e m o j i h ...
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I'...	o m g a g a i m s o o o i m g u n n a ...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! ...	i t h i n k m i b f i s c h e a t i ...

```
In [ ]: positive_words =' '.join([text for text in train['clean_text'][train['Sentiment'] == 1]])

        wordcloud = WordCloud(background_color = 'grey',width=800, height=500, random_state = 0, max_font_size = 110).generate(positive_wo
        plt.figure(figsize=(10, 7))
        plt.imshow(wordcloud, interpolation="bilinear")
        plt.axis('off')
        plt.title('The Positive Words after Cleaning')
        plt.show()
```

```
In [ ]: negative_words =' '.join([text for text in data['clean_text'][data['Sentiment'] == 0]])

        wordcloud = WordCloud(background_color = 'grey',width=800, height=500, random_state = 0, max_font_size = 110).generate(negative_wo
        plt.figure(figsize=(10, 7))
        plt.imshow(wordcloud, interpolation="bilinear")
        plt.axis('off')
        plt.title('The Negative Words after Cleaning ')
        plt.show()
```

```
In [13]: #Separates a file with mixed positive and negative examples into two.
         def separate_dataset(filename):
             good_out = open("good_file","w+",encoding="utf8");
             bad_out  = open("bad_file","w+",encoding="utf8");

             seen = 1;
             with open(filename,'r',encoding="utf8") as f:
                 reader = csv.reader(f)
                 next(reader)

                 for line in reader:
                     seen +=1
                     sentiment = line[1]
                     sentence = line[4]

                     if (sentiment == "0"):
                         bad_out.write(sentence+"\n")
                     else:
                         good_out.write(sentence+"\n")

                     if (seen%10000==0):
                         print (seen);

             good_out.close();
             bad_out.close();
```

```
In [15]: separate_dataset("Sentiment Analysis Dataset.csv");
```

```
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
110000
120000
130000
140000
150000
160000
170000
180000
190000
```

In [14]:
```
#Load Datafiles
def get_dataset(goodfile,badfile,limit,randomize=True):
    good_x = list(open(goodfile,"r",encoding="utf8").readlines())
    good_x = [s.strip() for s in good_x]

    bad_x  = list(open(badfile,"r",encoding="utf8").readlines())
    bad_x  = [s.strip() for s in bad_x]

    if (randomize):
        random.shuffle(bad_x)
        random.shuffle(good_x)

    good_x = good_x[:limit]
    bad_x = bad_x[:limit]

    x = good_x + bad_x
    x = [clean_str(s) for s in x]


    positive_labels = [[0, 1] for _ in good_x]
    negative_labels = [[1, 0] for _ in bad_x]
    y = np.concatenate([positive_labels, negative_labels], 0)
    return [x,y]
```

In [ ]:
```

#Generate random batches

def gen_batch(data, batch_size, num_epochs, shuffle=True):
    """
    Generates a batch iterator for a dataset.
    """
    data = np.array(data)
    data_size = len(data)
    num_batches_per_epoch = int((len(data)-1)/batch_size) + 1
    for epoch in range(num_epochs):
        # Shuffle the data at each epoch
        if shuffle:
            shuffle_indices = np.random.permutation(np.arange(data_size))
            shuffled_data = data[shuffle_indices]
        else:
            shuffled_data = data
        for batch_num in range(num_batches_per_epoch):
            start_index = batch_num * batch_size
            end_index = min((batch_num + 1) * batch_size, data_size)
            yield shuffled_data[start_index:end_index]
```

In [27]:
```
# Data Preparation
filename = "Sentiment Analysis Dataset.csv"
goodfile = "good_file"
badfile = "bad_file"
```

In [28]:
```
x_text, y = get_dataset(goodfile, badfile, 5000)
```

In [23]:
```
good_tweets = pd.read_csv('good_file',error_bad_lines=False)
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)

b'Skipping line 697294: expected 1 fields, saw 2\nSkipping line 697303: expected 1 fields, saw 2\nSkipping line 697308: expect
ed 1 fields, saw 4\nSkipping line 697309: expected 1 fields, saw 2\nSkipping line 697311: expected 1 fields, saw 2\nSkipping l
ine 697316: expected 1 fields, saw 2\nSkipping line 697321: expected 1 fields, saw 2\nSkipping line 697323: expected 1 fields,
saw 2\nSkipping line 697324: expected 1 fields, saw 2\nSkipping line 697326: expected 1 fields, saw 2\nSkipping line 697328: e
xpected 1 fields, saw 2\nSkipping line 697331: expected 1 fields, saw 2\nSkipping line 697333: expected 1 fields, saw 2\nSkipp
ing line 697336: expected 1 fields, saw 2\nSkipping line 697345: expected 1 fields, saw 2\nSkipping line 697351: expected 1 fi
elds, saw 2\nSkipping line 697357: expected 1 fields, saw 3\nSkipping line 697358: expected 1 fields, saw 3\nSkipping line 697
362: expected 1 fields, saw 2\nSkipping line 697364: expected 1 fields, saw 2\nSkipping line 697367: expected 1 fields, saw 2
\nSkipping line 697374: expected 1 fields, saw 2\nSkipping line 697375: expected 1 fields, saw 3\nSkipping line 697377: expect
```

In [24]:
```
print(good_tweets.shape)
```

```
(594585, 1)
```

```
In [25]: good_tweets.head(10)
```

```
Out[25]:
                          omg its already 7:30 :O
0            Juuuuuuuuuuuuuuuuussssst Chillin!!
1           handed in my uniform today . i miss you ...
2       hmmmm.... i wonder how she my number @-)
3           thanks to all the haters up in my face a...
4         Feeling strangely fine. Now I'm gonna go l...
5        You're the only one who can see this cause...
6                      uploading pictures on friendster
7           (: !!!!!! - so i wrote something last week. ...
8                             ... health class (what a joke!)
9         @ginaaa &lt;3 GO TO THE SHOW TONIGHT
```

```
In [ ]: bad_tweets = pd.read_csv('bad_file',error_bad_lines=False)
```

```
In [ ]: print(bad_tweets.shape)
```

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

```
In [ ]: !pip install tensorflow==1.14
        import tensorflow as tf
```

Model Building

In [ ]:
```python
import numpy as np
from IPython import embed

class CNN_LSTM(object):
    def __init__(self, sequence_length, num_classes, vocab_size, embedding_size, filter_sizes, num_filters, l2_reg_lambda=0.0,num_

        # PLACEHOLDERS
        self.input_x = tf.placeholder(tf.int32, [None, sequence_length], name="input_x")    # X - The Data
        self.input_y = tf.placeholder(tf.float32, [None, num_classes], name="input_y")      # Y - The Lables
        self.dropout_keep_prob = tf.placeholder(tf.float32, name="dropout_keep_prob")       # Dropout


        l2_loss = tf.constant(0.0) # Keeping track of l2 regularization loss

        #1. EMBEDDING LAYER ################################################################
        with tf.device('/cpu:0'), tf.name_scope("embedding"):
            self.W = tf.Variable(tf.random_uniform([vocab_size, embedding_size], -1.0, 1.0),name="W")
            self.embedded_chars = tf.nn.embedding_lookup(self.W, self.input_x)
            self.embedded_chars_expanded = tf.expand_dims(self.embedded_chars, -1)

        #2. CONVOLUTION LAYER + MAXPOOLING LAYER (per filter) ###############################
        pooled_outputs = []
        for i, filter_size in enumerate(filter_sizes):
            with tf.name_scope("conv-maxpool-%s" % filter_size):
                # CONVOLUTION LAYER
                filter_shape = [filter_size, embedding_size, 1, num_filters]
                W = tf.Variable(tf.truncated_normal(filter_shape, stddev=0.1), name="W")
                b = tf.Variable(tf.constant(0.1, shape=[num_filters]), name="b")
                conv = tf.nn.conv2d(self.embedded_chars_expanded, W,strides=[1, 1, 1, 1],padding="VALID",name="conv")
                # NON-LINEARITY
                h = tf.nn.relu(tf.nn.bias_add(conv, b), name="relu")
                # MAXPOOLING
                pooled = tf.nn.max_pool(h, ksize=[1, sequence_length - filter_size + 1, 1, 1], strides=[1, 1, 1, 1], padding='VAL
                pooled_outputs.append(pooled)

        # COMBINING POOLED FEATURES
        num_filters_total = num_filters * len(filter_sizes)
        self.h_pool = tf.concat(pooled_outputs, 3)
        self.h_pool_flat = tf.reshape(self.h_pool, [-1, num_filters_total])

        #3. DROPOUT LAYER ###################################################################
        with tf.name_scope("dropout"):
             self.h_drop = tf.nn.dropout(self.h_pool_flat, self.dropout_keep_prob)

        #4. LSTM LAYER ######################################################################
        cell = tf.contrib.rnn.LSTMCell(num_hidden,state_is_tuple=True)
        self.h_drop_exp = tf.expand_dims(self.h_drop,-1)
        val,state = tf.nn.dynamic_rnn(cell,self.h_drop_exp,dtype=tf.float32)

        #embed()

        val2 = tf.transpose(val, [1, 0, 2])
        last = tf.gather(val2, int(val2.get_shape()[0]) - 1)

        out_weight = tf.Variable(tf.random_normal([num_hidden, num_classes]))
        out_bias = tf.Variable(tf.random_normal([num_classes]))

        with tf.name_scope("output"):
            #lstm_final_output = val[-1]
            #embed()
            self.scores = tf.nn.xw_plus_b(last, out_weight,out_bias, name="scores")
            self.predictions = tf.nn.softmax(self.scores, name="predictions")

        with tf.name_scope("loss"):
            self.losses = tf.nn.softmax_cross_entropy_with_logits(logits=self.scores,labels=self.input_y)
            self.loss = tf.reduce_mean(self.losses, name="loss")

        with tf.name_scope("accuracy"):
            self.correct_pred = tf.equal(tf.argmax(self.predictions, 1),tf.argmax(self.input_y, 1))
            self.accuracy = tf.reduce_mean(tf.cast(self.correct_pred, "float"),name="accuracy")

        print ("(!) LOADED CNN-LSTM! :)")
        #embed()
```

In [ ]:
```python
import random
import sys
import os

file_name = "Sentiment Analysis Dataset.csv"
count = 1000

subscript = 1

while os.path.isfile('./good' + str(count) + '_' + str(subscript)):
    subscript += 1

t_file = list(open(file_name, 'r',encoding="utf8"))
good_file = open("good" + str(count) + '_' + str(subscript), 'a',encoding="utf8")
bad_file = open("bad" + str(count) + '_' +  str(subscript), 'a',encoding="utf8")

print("Opened file")

good_count = 0
bad_count = 0

while True:
    line = random.choice(t_file)
    line_split = line.split(',', 2)
    label = int(line_split[1])
    if label and good_count < count:
        good_file.write(line)
        good_count += 1
    elif not label and bad_count < count:
        bad_file.write(line)
        bad_count += 1
    elif bad_count >= count and good_count >= count:
        break
```

In [ ]:
```python
import numpy as np
import time
import datetime
from tensorflow.contrib import learn

from IPython import embed

# Parameters
dev_size = .10

# Model Hyperparameters
embedding_dim  = 32     #128
max_seq_legth = 70
filter_sizes = [3,4,5]  #3
num_filters = 32
dropout_prob = 0.5 #0.5
l2_reg_lambda = 0.0
use_glove = True #Do we use glove

# Training parameters
batch_size = 128
num_epochs = 10 #200
evaluate_every = 100 #100
checkpoint_every = 100000 #100
num_checkpoints = 1 #Checkpoints to store

# Misc Parameters
allow_soft_placement = True
log_device_placement = False


# Data Preparation
filename = "Sentiment Analysis Dataset.csv"
goodfile = "good_file"
badfile = "bad_file"

# Load data
print("Loading data...")
x_text, y = get_dataset(goodfile, badfile, 5000)

#TODO: MAX LENGTH
# Build vocabulary
max_document_length = max([len(x.split(" ")) for x in x_text])
vocab_processor = learn.preprocessing.VocabularyProcessor(max_document_length)
x = np.array(list(vocab_processor.fit_transform(x_text)))

# Randomly shuffle data
np.random.seed(42)
shuffle_indices = np.random.permutation(np.arange(len(y)))
x_shuffled = x[shuffle_indices]
y_shuffled = y[shuffle_indices]

# Split train/test set
# TODO: This is very crude, should use cross-validation
dev_sample_index = -1 * int(dev_size * float(len(y)))
x_train, x_test = x_shuffled[:dev_sample_index], x_shuffled[dev_sample_index:]
y_train, y_test = y_shuffled[:dev_sample_index], y_shuffled[dev_sample_index:]
print("Vocabulary Size: {:d}".format(len(vocab_processor.vocabulary_)))
print("Train/Test split: {:d}/{:d}".format(len(y_train), len(y_test)))

#embed()
```

Training and Evaluation

In [ ]:

```
with tf.Graph().as_default():
    session_conf = tf.ConfigProto(allow_soft_placement=True, log_device_placement=False)
    sess = tf.Session(config=session_conf)
    with sess.as_default():
        #embed()
        cnn = CNN_LSTM(x_train.shape[1],y_train.shape[1],len(vocab_processor.vocabulary_),embedding_dim,filter_sizes,num_filters,

        # Define Training procedure
        global_step = tf.Variable(0, name="global_step", trainable=False)
        optimizer = tf.train.AdamOptimizer(1e-3)
        grads_and_vars = optimizer.compute_gradients(cnn.loss)
        train_op = optimizer.apply_gradients(grads_and_vars, global_step=global_step)


        # Output directory for models and summaries
        timestamp = str(int(time.time()))
        out_dir = os.path.abspath(os.path.join(os.path.curdir, "runs", timestamp))
        print("Writing to {}\n".format(out_dir))

        # Summaries for loss and accuracy
        loss_summary = tf.summary.scalar("loss", cnn.loss)
        acc_summary = tf.summary.scalar("accuracy", cnn.accuracy)

        # Train Summaries
        train_summary_op = tf.summary.merge([loss_summary, acc_summary, grad_summaries_merged])
        train_summary_dir = os.path.join(out_dir, "summaries", "train")
        train_summary_writer = tf.summary.FileWriter(train_summary_dir, sess.graph)

        # Dev summaries
        dev_summary_op = tf.summary.merge([loss_summary, acc_summary])
        dev_summary_dir = os.path.join(out_dir, "summaries", "dev")
        dev_summary_writer = tf.summary.FileWriter(dev_summary_dir, sess.graph)

        # Checkpoint directory. Tensorflow assumes this directory already exists so we need to create it
        checkpoint_dir = os.path.abspath(os.path.join(out_dir, "checkpoints"))
        checkpoint_prefix = os.path.join(checkpoint_dir, "model")
        if not os.path.exists(checkpoint_dir):
            os.makedirs(checkpoint_dir)
        saver = tf.train.Saver(tf.global_variables(), max_to_keep=num_checkpoints)

        # Write vocabulary
        vocab_processor.save(os.path.join(out_dir, "vocab"))

        # Initialize all variables
        sess.run(tf.global_variables_initializer())

        #TRAINING STEP
        def train_step(x_batch, y_batch,save=False):
            feed_dict = {
              cnn.input_x: x_batch,
              cnn.input_y: y_batch,
              cnn.dropout_keep_prob: dropout_prob
            }
            _, step, summaries, loss, accuracy = sess.run(
                [train_op, global_step, train_summary_op, cnn.loss, cnn.accuracy],
                feed_dict)
            time_str = datetime.datetime.now().isoformat()
            print("{}: step {}, loss {:g}, acc {:g}".format(time_str, step, loss, accuracy))
            if save:
                train_summary_writer.add_summary(summaries, step)

        #EVALUATE MODEL
        def test_step(x_batch, y_batch, writer=None,save=False):
            feed_dict = {
              cnn.input_x: x_batch,
              cnn.input_y: y_batch,
              cnn.dropout_keep_prob: 0.5
            }
            step, summaries, loss, accuracy = sess.run(
                [global_step, dev_summary_op, cnn.loss, cnn.accuracy],
                feed_dict)
            time_str = datetime.datetime.now().isoformat()
            print("{}: step {}, loss {:g}, acc {:g}".format(time_str, step, loss, accuracy))
            if save:
                if writer:
                    writer.add_summary(summaries, step)
```

In [ ]:
```
#CREATE THE BATCHES GENERATOR
batches = gen_batch(list(zip(x_train, y_train)), batch_size, num_epochs)

#TRAIN FOR EACH BATCH
for batch in batches:
    x_batch, y_batch = zip(*batch)
    train_step(x_batch, y_batch)
    current_step = tf.train.global_step(sess, global_step)
    if current_step % evaluate_every == 0:
        print("\nEvaluation:")
        test_step(x_test, y_test, writer=dev_summary_writer)
          print("")
    if current_step % checkpoint_every == 0:
        path = saver.save(sess, checkpoint_prefix, global_step=current_step)
        print("Saved model checkpoint to {}\n".format(path))
```

In [ ]:
```
```