Sentiment Analysis using Neural Network (CNN)

In [31]:

```
"""Author: Siddhant Shrivastava
<siddhant.shrivastava23@gmail.com>\nSentiment Analysis using word2vec, NN.
\
NLP Project; Siddhant Shrivastava, Aditya Srivastava, Pranav Nair; Monsoon
2017, IIIT-Hyderabad"""
print(__doc__)
```

```
Author: Siddhant Shrivastava <siddhant.shrivastava23@gmail.com>
Sentiment Analysis using word2vec, NN. NLP Project; Siddhant Shrivastava, A
ditya Srivastava, Pranav Nair; Monsoon 2017, IIIT-Hyderabad
```

Reference: Sentiment analysis on Twitter using word2vec and keras by Ahmed Besbes

Import Modules

In [2]:

```
import pandas as pd
pd.options.mode.chained_assignment = None
import numpy as np
from copy import deepcopy
from string import punctuation
from random import shuffle

import gensim
from gensim.models.word2vec import Word2Vec # the word2vec model gensim cla
ss
LabeledSentence = gensim.models.doc2vec.LabeledSentence

from tqdm import tqdm
tqdm.pandas(desc="progress-bar")

from nltk.tokenize import TweetTokenizer
tokenizer = TweetTokenizer()

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
D:\siddh\Software\anaconda\anaconda3\lib\site-packages\gensim\utils.py:860:
UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

Define Functions

Function to load the dataset and extract the columns we need

In [3]:

```
def ingest():
    """Load dataset, extract the sentiment and tweet's text columns"""
    data = pd.read_csv('../../dataset/cnn_dataset/tweets.csv', encoding="ISO-8859-1")
    # data.drop(['ItemID', 'Date', 'Blank', 'SentimentSource'], axis=1, inplace=True)
    data.drop(['ItemID', 'SentimentSource'], axis=1, inplace=True)
    data = data[data.Sentiment.isnull() == False]
    data['Sentiment'] = data['Sentiment'].map(int)
    data = data[data['SentimentText'].isnull() == False]
    data.reset_index(inplace=True)
    data.drop('index', axis=1, inplace=True)
    data['Sentiment'] = data['Sentiment'].map({4:1, 0:0})
    print('dataset loaded with shape: ' + str(data.shape))
    return data
```

Tokenizing function

Splits each tweet into tokens and removes user mentions, hashtags and urls as they do not provide enough semantic information for the task

In [4]:

```
def tokenize(tweet):
    try:
        # tweet = unicode(tweet.decode('utf-8').lower())
        # tweet = unicode(tweet.decode('latin-1').lower())
        tweet = tweet.lower()
        tokens = tokenizer.tokenize(tweet)
        # tokens = filter(lambda t: not t.startswith('@'), tokens)
        # tokens = filter(lambda t: not t.startswith('#'), tokens)
        # tokens = filter(lambda t: not t.startswith('http'), tokens)
        return tokens
    except:
        return 'NC'
```

Process tokenized data

Tokenization results should now be cleaned to remove lines with 'NC', resulting from a tokenization error

In [5]:

```
def postprocess(data, n=1000000):
    data = data.head(n)
    data['tokens'] = data['SentimentText'].progress_map(tokenize)  ## progress_map is a variant of the map function plus a progress bar. Handy to monitor DataFrame creations.
    print("Tokenization done")
    print(data.head(5))
    # print(data.tokens.value_counts())
    data = data[data.tokens != 'NC']
    data.reset_index(inplace=True)
```

```
    data.drop('index', inplace=True, axis=1)
    return data
```

Function to turn tokens to LabeledSentence objects before feeding to the word2vec model

In [6]:

```
def labelizeTweets(tweets, label_type):
    labelized = []
    for i,v in tqdm(enumerate(tweets)):
        label = '%s_%s'%(label_type,i)
        labelized.append(LabeledSentence(v, [label]))
    return labelized
```

Function to create averaged tweet vector

In [7]:

```
def buildWordVector(tokens, size):
    """Given a list of tweet tokens, creates an averaged tweet vector."""
    vec = np.zeros(size).reshape((1, size))
    count = 0.
    for word in tokens:
        try:
            vec += tweet_w2v[word].reshape((1, size)) * tfidf[word]
            count += 1.
        except KeyError: # handling the case where the token is not
                         # in the corpus. useful for testing.
            continue
    if count != 0:
        vec /= count
    return vec
```

Load and Process Data

In [8]:

```
data = ingest()
data.head(5)
```

```
dataset loaded with shape: (1600000, 4)
```

Out[8]:

	Sentiment	Date	Blank	SentimentText
0	0	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	is upset that he can't update his Facebook by ...
2	0	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	@Kenichan I dived many times for the ball. Man...

	Sentiment	Date	Blank	SentimentText
3	0	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	my whole body feels itchy and like its on fire
4	0	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	@nationwideclass no, it's not behaving at all....

In [9]:

```
data.Sentiment.value_counts()
# {'0': "negative sentiment", '1': "positive sentiment"}
```

Out[9]:

```
1    800000
0    800000
Name: Sentiment, dtype: int64
```

Tokenize and clean data

In [10]:

```
data = postprocess(data)
```

```
progress-bar: 100%|
|| 1000000
/1000000 [01:06<00:00, 14972.83it/s]
```

```
Tokenization done
   Sentiment                          Date     Blank  \
0          0  Mon Apr 06 22:19:45 PDT 2009  NO_QUERY
1          0  Mon Apr 06 22:19:49 PDT 2009  NO_QUERY
2          0  Mon Apr 06 22:19:53 PDT 2009  NO_QUERY
3          0  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY
4          0  Mon Apr 06 22:19:57 PDT 2009  NO_QUERY

                                       SentimentText  \
0  @switchfoot http://twitpic.com/2y1zl - Awww, t...
1  is upset that he can't update his Facebook by ...
2  @Kenichan I dived many times for the ball. Man...
3    my whole body feels itchy and like its on fire
4  @nationwideclass no, it's not behaving at all....

                                              tokens
0  [@switchfoot, http://twitpic.com/2y1zl, -, aww...
1  [is, upset, that, he, can't, update, his, face...
2  [@kenichan, i, dived, many, times, for, the, b...
3  [my, whole, body, feels, itchy, and, like, its...
4  [@nationwideclass, no, ,, it's, not, behaving,...
```

We are considering 1,000,000 (1 million) records.

In [11]:

```
data.shape
```

Out[11]:

```
(1000000, 5)
```

In [12]:

```
n = 1000000
```

Build the word2vec model

Define the training and test dataset

In [13]:

```
x_train, x_test, y_train, y_test = train_test_split(np.array(data.head(n).t
okens),
                                                    np.array(data.head(n).Se
timent), test_size=0.2)
```

Turn tokens into LabeledSentence Object

Before feeding lists of tokens into the word2vec model, we must turn them into LabeledSentence objects beforehand.

In [14]:

```
x_train.shape
```

Out[14]:

```
(800000,)
```

In [15]:

```
x_train = labelizeTweets(x_train, 'TRAIN')
x_test = labelizeTweets(x_test, 'TEST')
```

```
800000it [00:04, 164882.77it/s]
200000it [00:00, 293336.49it/s]
```

In [16]:

```
print(x_train[0])
```

```
LabeledSentence(['feels', 'she', 'doesnt', 'have', 'enough', 'clothes', 'ye
t', 'her', 'bag', 'is', 'already', 'overflowing', '.', "don't", 'know', 'wh
at', 'to', 'bring', 'and', 'what', '...', 'http://plurk.com/p/yjyrp'], ['TR
AIN_0'])
```

Build the word2vec model from x_train i.e. the corpus.

Set the number of dimensions of the vector space

In [17]:

```
n_dim = 200
```

In [18]:

```
tweet_w2v = Word2Vec(size=n_dim, min_count=10)
tweet_w2v.build_vocab([x.words for x in tqdm(x_train)])
tweet_w2v.train([x.words for x in tqdm(x_train)],total_examples=tweet_w2v.c
orpus_count, epochs=tweet_w2v.iter)
```

```
100%|
| 800000/800000 [00:00<00:00, 1631538.46it/s]
100%|
| 800000/800000 [00:00<00:00, 1590971.49it/s]
```

Out[18]:

```
43495480
```

Check semantic realatioship set by word2vec

In [19]:

```
tweet_w2v.most_similar('good')
```

Out[19]:

```
[('goood', 0.6968518495559692),
 ('great', 0.6915127038955688),
 ('pleasant', 0.6437491178512573),
 ('tough', 0.6366854906082153),
 ('nice', 0.6066265106201172),
 ('gd', 0.6049803495407104),
 ('goooood', 0.6010672450065613),
 ('terrible', 0.5997786521911621),
 ('rough', 0.5993411540985107),
 ('bad', 0.5941449999809265)]
```

Build the Sentiment Classifier

Build the tf-idf matrix

to compute the tf-idf score which is a weighted average where each weight gives the importance of the word with respect to the corpus.

In [21]:

```
print('building tf-idf matrix ...')
vectorizer = TfidfVectorizer(analyzer=lambda x: x, min_df=10)
matrix = vectorizer.fit_transform([x.words for x in x_train])
tfidf = dict(zip(vectorizer.get_feature_names(), vectorizer.idf_))
print('vocab size : %s' % (len(tfidf)))
```

```
building tf-idf matrix ...
vocab size : 25737
```

Convert x_train and x_test to a list of vectors

Also scale each column to have zero mean and unit standard deviation.

In [22]:

```
from sklearn.preprocessing import scale
train_vecs_w2v = np.concatenate([buildWordVector(z, n_dim) for z in tqdm(ma
p(lambda x: x.words, x_train))])
train_vecs_w2v = scale(train_vecs_w2v)

test_vecs_w2v = np.concatenate([buildWordVector(z, n_dim) for z in tqdm(map
(lambda x: x.words, x_test))])
test_vecs_w2v = scale(test_vecs_w2v)
```

```
800000it [01:51, 7189.78it/s]
200000it [00:23, 8554.25it/s]
```

Feed vectors into Neural Network Classifier

In [28]:

```
from keras.layers.core import Dense, Activation, Dropout
from keras.models import Sequential
model = Sequential()
model.add(Dense(32, activation='relu', input_dim=200))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(train_vecs_w2v, y_train, epochs=9, batch_size=32, verbose=2)
```

```
Epoch 1/9
23s - loss: 0.3494 - acc: 0.8538
Epoch 2/9
22s - loss: 0.3368 - acc: 0.8594
Epoch 3/9
21s - loss: 0.3331 - acc: 0.8610
Epoch 4/9
21s - loss: 0.3309 - acc: 0.8622
Epoch 5/9
21s - loss: 0.3297 - acc: 0.8625
Epoch 6/9
21s - loss: 0.3286 - acc: 0.8631
Epoch 7/9
22s - loss: 0.3277 - acc: 0.8636
Epoch 8/9
21s - loss: 0.3271 - acc: 0.8640
Epoch 9/9
21s - loss: 0.3266 - acc: 0.8640
```

Out[28]:

```
<keras.callbacks.History at 0x20718e99ba8>
```

Evaluate on test set

In [29]:

```
score = model.evaluate(test_vecs_w2v, y_test, batch_size=128, verbose=2)
print(score[1])
```

```
0.85963
```

Accuracy obtained = 85.963% (~86%)

Save neural net

Save and Load Keras Deep Learning models

In [30]:

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model.h5")
print("Saved model to disk")
```

```
Saved model to disk
```