Sentiment Analysis

Neural Network with pretrained vectors

Prepare the tweets

1. Load tweets and labels in pandas dataframes
2. Stem and clean tweets
3. Finish with final tweets and labels in pandas dataframes

If train_.. is 1, then train that model, if 0 then do not train that model

if eval_.. is 1, then load that model and evaluate on train and test data, if 0 do not load or evaluate

Note: if you want to fully train the model, both train.. *and eval*.. should be set to 1

In [1]:

```
train_NN = 0
eval_NN = 1
train_Linear = 0
eval_Linear = 1
```

In [2]:

```
import pandas as pd
import nltk
import string
import tensorflow as tf
from sklearn.model_selection import train_test_split
import os
import numpy as np
```

In [3]:

```
labels = pd.read_csv('Tweets.csv', usecols = ['airline_sentiment'])
labels = labels.replace(['negative', 'neutral', 'positive'], [0, 1, 2])
tweets = pd.read_csv('Tweets.csv', usecols = ['text'])
```

In [4]:

```
# create custom stoplist
stoplist = ['to', 'the', 'i', 'a', 'and', 'you', 'is', 'my', 'that',
            'was', 'but', 'me', 'your', 'this', 'we', 'are', 'so', 'be',
            'with', 'he', 'she', 'them', 'it', 'their', 'at', 'get', 'in',
            'there', 'on', 'have', 'am', 'when', 'if', '...', 'do', 'of',
            'or', 'tho', 'though', 'for', 'from', 'u', "i'm", 'because',
            'us', 'an', 'just', 'by', 'had', 'all', 'now', 'will']
#stoplist = ['to', 'the', 'i', 'a', 'me', 'your', 'he', 'she', '...']
#stoplist = []
```

In [5]:

```
def lem(tweet, tknzr, stoplist):
    temp = tknzr.tokenize(tweet)
    result = []
    tweet_length = 0
    for x in temp:
        if x in string.punctuation:
            continue
        elif x in ['“', '”', "’", ' ']:
            continue
        elif str('\n') in x:
            continue
        elif x in stoplist:
            continue
        else:
            result.append(x)
            tweet_length += 1
    if len(result) == 0:
        result.append('0')
    r = ' '.join(result)
    return r, tweet_length
```

In [6]:

```
# create tokenizer
tknzr = nltk.tokenize.casual.TweetTokenizer(preserve_case = False,
strip_handles = True, reduce_len = True)
count = 0
vals = tweets.text.values
tweet_length_max = 0
with open('all_text.txt', 'w', encoding = 'utf-8') as f:
    for x in vals:
        tweet = x
        r, tweet_length = lem(tweet, tknzr, stoplist)
        f.write(r + '\n')
        count += 1
        if tweet_length > tweet_length_max:
            tweet_length_max = tweet_length
f.close()
print('maximum length of tweet: ', tweet_length_max)
```

```
maximum length of tweet:  26
```

In [7]:

```
cols = ['word' + str(x) for x in range(0, tweet_length_max)]
tweets = pd.read_csv('all_text.txt', header = None, delim_whitespace = True
,
                     names = cols)
tweets = tweets.fillna('0')
print(len(tweets))
```

```
14640
```

So now we have our tweets and labels in DataFrames labeled tweets, and labels respectively

Prepare embedding layer

Will use pre-trained GloVe vectors, trained on tweets and downloaded from https://nlp.stanford.edu/projects/glove/ The raw text file is loaded, and then transformed into a dictionary. This dictionary can take in a word and will return the GloVe vectorization for that word. Each word is a 50 dimension vector in this model

In [8]:

```
import numpy as np
```

In [9]:

```
def loadGloveModel(gloveFile):
    print("Loading Glove Model")
    f = open(gloveFile,'r')
    model = {}
    for line in f:
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print("Done.",len(model)," words loaded!")
    return model
```

In [10]:

```
embed_model = loadGloveModel('glove_twitter_27B_50d.txt')
dim_length = 50
```

```
Loading Glove Model
Done. 1193514  words loaded!
```

Neural Network

Neural Network Training

1. Split tweets and labels into testing and training data
2. Model traning input a. embed(tweets), takes the raw tweets and returns the vector representation. This is fed directly to model b. input_fn_train transforms the input so tensorflow can understand it
3. Training the model a. use a precanned tensorflow DNNClassifier, train for 100000 iterations

In [11]:

```
x_train, x_test, y_train, y_test = train_test_split(tweets, labels,
test_size = .2, random_state = 42)
```

In [12]:

```
def embed(data, embed_model, dim_length):
    zero_val = [0 for y in range(dim_length)]
    data_mat = data.as_matrix()
    x_embed = []
    for tweet in data_mat:
        temp = []
        temp = []
        for word in tweet:
            if word in embed_model:
                temp.append(embed_model[word])
            else:
                temp.append(zero_val)
        x_embed.append(temp)
    return np.asarray(x_embed)
```

In [13]:

```
def input_fn_train(x, y, col):
    y_data = y.values.flatten()
    x_data = {}
    for c in range(len(col)):
        x_data[col[c]] = x[:, c]
    return tf.estimator.inputs.numpy_input_fn(x=x_data,
                                              y=y_data,
                                              num_epochs=None,
                                              shuffle=False)
```

In [14]:

```
import os
cwd = os.getcwd()
NN_model_directory = os.path.join(cwd, 'NN_model_directory')
if not os.path.exists(NN_model_directory):
    os.makedirs(NN_model_directory)
```

In [15]:

```
if eval_NN == 1:
    features = [tf.contrib.layers.real_valued_column(x, dimension =
dim_length) for x in cols]
    optimizer = tf.train.AdadeltaOptimizer(learning_rate = 0.001,
                                           rho=0.95,
                                           epsilon=1e-08,
                                           use_locking=False,
                                           name='Adadelta')
    m = tf.estimator.DNNClassifier(model_dir = NN_model_directory,
                                   feature_columns = features,
                                   hidden_units = [100, 80],
                                   n_classes = 3,
                                   optimizer = optimizer)
```

```
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_model_dir':
'/media/michael/1E72EDB672ED9337/airline_twitter/models_pretrained_vectors/N
odel_directory', '_tf_random_seed': 1, '_save_summary_steps': 100, '_save_c
heckpoints_secs': 600, '_save_checkpoints_steps': None, '_session_config':
None, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '
_log_step_count_steps': 100}
```

In [16]:

```
if train_NN == 1:
    x = embed(x_train, embed_model, dim_length)
    y = y_train
    m.train(input_fn = input_fn_train(x, y, cols),
            steps = 100000)
```

Neural Network Evaluation

No we evaluate the model on the training data and the testing data

In [17]:

```
def input_fn_eval(x, y, col):
    y_data = y.values.flatten()
    x_data = {}
    for c in range(len(col)):
        x_data[col[c]] = x[:, c]
    return tf.estimator.inputs.numpy_input_fn(x=x_data,
                                               y=y_data,
                                               num_epochs = 1,
                                               shuffle=False)
```

In [18]:

```
if eval_NN == 1:
    x = embed(x_train, embed_model, dim_length)
    y = y_train
    train_eval = m.evaluate(input_fn = input_fn_eval(x, y, cols))
    print('accuracy of training model: ', train_eval['accuracy'])
```

```
INFO:tensorflow:Starting evaluation at 2017-09-21-00:27:18
INFO:tensorflow:Restoring parameters from
/media/michael/1E72EDB672ED9337/airline_twitter/models_pretrained_vectors/N
del_directory/model.ckpt-100002
INFO:tensorflow:Finished evaluation at 2017-09-21-00:27:18
INFO:tensorflow:Saving dict for global step 100002: accuracy = 0.754952, av
erage_loss = 0.608606, global_step = 100002, loss = 77.4782
accuracy of training model:  0.754952
```

In [19]:

```
if eval_NN == 1:
    x = embed(x_test, embed_model, dim_length)
    y = y_test
    test_eval = m.evaluate(input_fn = input_fn_eval(x, y, cols))
    print('accuracy of testing model: ', test_eval['accuracy'])
```

```
INFO:tensorflow:Starting evaluation at 2017-09-21-00:27:19
INFO:tensorflow:Restoring parameters from
/media/michael/1E72EDB672ED9337/airline_twitter/models_pretrained_vectors/N
del_directory/model.ckpt-100002
INFO:tensorflow:Finished evaluation at 2017-09-21-00:27:20
INFO:tensorflow:Saving dict for global step 100002: accuracy = 0.742145, av
erage_loss = 0.638584, global_step = 100002, loss = 81.2945
accuracy of testing model:  0.742145
```

So: after 100000 iterations, I got 74% accuracy on our testing data

Linear Classifier

Train Linear Model

In [20]:

```
import os
cwd = os.getcwd()
Linear_model_directory = os.path.join(cwd, 'Linear_model_directory')
if not os.path.exists(Linear_model_directory):
    os.makedirs(Linear_model_directory)
```

In [21]:

```
if eval_Linear == 1:
    features = [tf.contrib.layers.real_valued_column(x, dimension =
dim_length) for x in cols]
    optimizer = tf.train.AdadeltaOptimizer(learning_rate = 0.001,
                                           rho=0.95,
                                           epsilon=1e-08,
                                           use_locking=False,
                                           name='Adadelta')
    m_Linear = tf.estimator.LinearClassifier(model_dir =
Linear_model_directory,
                                     feature_columns = features,
                                     n_classes = 3,
                                     optimizer = optimizer)
```

```
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_model_dir':
'/media/michael/1E72EDB672ED9337/airline_twitter/models_pretrained_vectors/L
ar_model_directory', '_tf_random_seed': 1, '_save_summary_steps': 100, '_sa
ve_checkpoints_secs': 600, '_save_checkpoints_steps': None,
'_session_config': None, '_keep_checkpoint_max': 5,
'_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps': 100}
```

In [22]:

```
if train_Linear == 1:
    x = embed(x_train, embed_model, dim_length)
    y = y_train
    m_Linear.train(input_fn = input_fn_train(x, y, cols),
                   steps = 100000)
```

Evaluate Linear Model

In [23]:

```
if eval_Linear == 1:
    x = embed(x_train, embed_model, dim_length)
    y = y_train
    train_eval = m_Linear.evaluate(input_fn = input_fn_eval(x, y, cols))
    print('accuracy of training model: ', train_eval['accuracy'])
```

```
INFO:tensorflow:Starting evaluation at 2017-09-21-00:27:21
INFO:tensorflow:Restoring parameters from
/media/michael/1E72EDB672ED9337/airline_twitter/models_pretrained_vectors/Li
```

```
r_model_directory/model.ckpt-100001
INFO:tensorflow:Finished evaluation at 2017-09-21-00:27:22
INFO:tensorflow:Saving dict for global step 100001: accuracy = 0.707223, av
erage_loss = 0.713475, global_step = 100001, loss = 90.8285
accuracy of training model:  0.707223
```

In [24]:

```
if eval_Linear == 1:
    x = embed(x_test, embed_model, dim_length)
    y = y_test
    test_eval = m_Linear.evaluate(input_fn = input_fn_eval(x, y, cols))
    print('accuracy of testing model: ', test_eval['accuracy'])
```

```
INFO:tensorflow:Starting evaluation at 2017-09-21-00:27:23
INFO:tensorflow:Restoring parameters from
/media/michael/1E72EDB672ED9337/airline_twitter/models_pretrained_vectors/Li
r_model_directory/model.ckpt-100001
INFO:tensorflow:Finished evaluation at 2017-09-21-00:27:24
INFO:tensorflow:Saving dict for global step 100001: accuracy = 0.719604, av
erage_loss = 0.703795, global_step = 100001, loss = 89.5962
accuracy of testing model:  0.719604
```

So after running the linear model for 100000 steps, I got a training accuracy of 72%