SENTIMENT ANALYSIS REPORT
(7th February 2015)

First of all, I have worked on **SentiWordNet v3.0.0**, whenever I encounter '#', I ignore those lines. Then, I split the lines on token '#', I ignore the ID and I took only the pos/term as a key and corresponding this key I store a pair of {Pos score, Neg score}. Thus a dictionary named sent_word_net is formed.

I have worked on **Amazon Product Review Dataset** although my project is Group1. Out of so many reviews I have worked on 1657 reviews.

I have used NLP(Natural language Processing) tool i.e. NLTK(for python) and used this on the amazon dataset. NLTK tool will give the pair of word and corresponding tag {'word', 'tag'}. With the help of that tag, the Key will be formed i.e. pos/ word.

Methodology:
From dictionary sent_word_net, I collect all the positive and negative scores corresponding keys for each review. Thus, Total Positive and Total Negative score for a line/review I get. Following this, I calculated the mean positive and mean negative scores and Objectivity score (1 - Pos score - Neg score) as well. The review is termed as positive(1) if the mean positive is greater equal to that mean negative otherwise it is termed as Negative(0). The score assigned as positive(1) and negative(0) is termed as ***Predicted Score***. The extracted review Id of each line and the corresponding predicted score is then added in a dictionary to maintain the various reviews and their score to be fetched at later point.

I did manual evaluation of 100 reviews which I stored in a Text file named ManualReview.txt containing *review ID* and the *manual score*(for ***positive: 1*** and for ***negative: 0***). This Manual review text file is read and each review is checked and its corresponding predicted score is retrieved from the dictionary.

The ***manual(Actual) score*** is matched with predicted score. Four cases formed:

1: ***True Positive:*** If predicted is equal to manual and predicted equals to positive, then the review is termed as true positive.
2: ***True Negative:*** If predicted is equal to manual and predicted equals to Negative, then the review is termed as true Negative.
3: ***False Positive:*** If predicted is not equal to manual and predicted equals to positive, then the review is termed as false positive.
4: ***False Negative:*** If predicted is not equal to manual and predicted equals to Negative, then the review is termed as false Negative.

Total TP, TN, FP, FN reviews are calculated and displayed. Also, the false positive and false negative review id's were collected and later reviewed.

Values of confidence matrix obtained are :

CONFIDENCE MATRIX

	Predicted +	Predicted -
Actual +	63	3
Actual -	29	5

TP: 63
TN: 5
FP: 29
FN: 3

Total True predicted: 68% ~ 70%
and False Predicted: ~30%

Therefore,
Accuracy Probability: ~0.7
Error Probability: ~0.3

Based on the review id's of FP and FN, I manually evaluated those Id's to find the reason for misclassification of those review ID's. The negative and positive score of FP and FN varies a little bit with less difference of around ~0.05 between them while those of True positive and True Negative the difference between their positive and negative values varies with greater difference. But some TP and TN's difference was like FP and FN which require comparing and checking each word of line which are present in sentiWordNet dictionary for 50/100 reviews.

****

Below is the code:

```
#! /usr/bin/env python

import numpy as np
import nltk;
import pprint;
import csv
import collections
import textwrap

from nltk.corpus import wordnet as wn
from nltk.tokenize import word_tokenize

inputLines = []
filePath = 'sentiword.txt'
wordnet = []
falseNeg = []
falsePos = []
truePos = []
trueNeg = []
global reviewDict
reviewDict = {}

#define two lists: one for positive score and other for negative
score
posList = []
negList = []

#variable for confidence matrix
global falsePositive
global falseNegative
global trueNegative
global truePositive

#loading senti word net dictionary
def sentiWordNet():

    sent_scores = collections.defaultdict(list)

    f = open(filePath)
    reader = csv.reader(f, delimiter='\t')
    for line in reader:
        if line[0].startswith("#"):
            continue
        if len(line) == 1:
            continue

        POS, ID, PosScore, NegScore, SynsetTerms, Gloss = line

        if len(POS) == 0 or len(ID) == 0:
            continue
        # print POS,PosScore,NegScore,SynsetTerms
        for term in SynsetTerms.split(" "):
```

```
            # drop #number at the end of every term
            term = term.split("#")[0]
            term = term.replace("-", " ").replace("_", " ")
            key = "%s/%s" % (POS, term.split("#")[0])
            sent_scores[key].append((float(PosScore),
float(NegScore)))

    for key, value in sent_scores.items():
        sent_scores[key] = np.mean(value, axis=0)

    return sent_scores

sentiLib = sentiWordNet()

#to determine whether the word is stopword in english dictionary or
not
def is_stopWord(word):
    if word.lower() in nltk.corpus.stopwords.words('english'):
        return True
    return False

#Read sentiwordNet and calculate scores
def sentiment(line):

    tokens = word_tokenize(line)
    tag_tuples = nltk.pos_tag(tokens)
    tag_type = 0

    for (string, tag) in tag_tuples:

        if tag.startswith("JJ"):
            tag_type = "a"
        if tag.startswith("NN"):
            tag_type = "n"
        if tag.startswith("RB"):
            tag_type = "r"
        if tag.startswith("VB"):
            tag_type = "v"

        #remove stop words
        if is_stopWord(string):
            continue

        #remove punctuations
        if not is_punctuation(string):
            token = {'word':string, 'pos':tag_type}
            sentence = "%s/%s"%(tag_type, string)

            if sentence in sentiLib:
                pos, neg = sentiLib[sentence]
                posList.append(pos)
                negList.append(neg)
```

```
#check whether parametric value is punctuation or not
def is_punctuation(string):
    for char in string:
        if char.isalpha() or char.isdigit():
            return False
    return True

#Function to read input file
def readFileContent():
    print 'Inside readFileContent Method'

    LinePos = 0
    lineNeg = 0
    lineCount = 0
    objScore = 0
    ch = 0

    print 'PREDICTED SCORE:'
    print 'LineNum\t\tReview Id\t\tTitle\t\t\tPositive Score\t
\tNegative Score\t\tObj Score\t\t\tResult'

    with open('randomfile2.txt') as inputfile:
        for line in inputfile:
            if (ch > 0 and ch <= 300):
                lineCount = lineCount + 1

                fileDesc = line.split('\t')

                inputLines.append(line)
                sentiment(line)

                #calculate total positive score for a line
                LinePos = calculateMean(posList)
                lineNeg = calculateMean(negList)
                objScore = 1 - LinePos - lineNeg

                predScore = PosNeg(LinePos, lineNeg)
                #dictProgramReview(fileDesc[0], predScore)
                reviewDict[str(fileDesc[0])] = str(predScore)

                del posList[:]
                del negList[:]

                print str(lineCount) + '\t\t' + str(fileDesc[0]) +
'\t\t' + str(fileDesc[1]) + '\t\t' + str(LinePos) + '\t\t' +
str(lineNeg) + '\t\t' + str(objScore)  + '\t\t' +
str(PosNeg(LinePos, lineNeg))

                ch = ch + 1
            else:
                ch = ch + 1

    print 'going inside readManualFile'
```

```
    tp, tn, fp, fn = readManualFile()
    print 'True positive: ' , str(tp), '\n', 'True Negative',
str(tn)
    print 'False positive: ', str(fp), '\n', 'False Negative',
str(fn)

#function to calculate more Positive or more negative
def PosNeg(posNum, negNum):
    if posNum >= negNum:
        return 1

    else:
        return 0

#function to calculate mean of a list
def calculateMean(list):
    return(reduce(lambda x, y: x + y, list) / len(list))

#Manually read a file to get the review
def readManualFile():

    falsePositive = 0
    falseNegative = 0
    trueNegative = 0
    truePositive = 0

    f = open('ManualReview.txt')

    for line in range(100):
        id,scoreMan = f.readline().rsplit(None, 1)
        scorePred = reviewDict[str(id)]

        #Calculate confidence matrix values
        print 'score pred: ',str(scorePred), 'scoreman: ',
str(scoreMan)
        if str(scorePred) == str(0):
            if str(scorePred) == str(scoreMan):
                print 'true negative', str(trueNegative), 'predicted
score: ' , str(scorePred), 'Manual score: ' , str(scoreMan)
                trueNegative = trueNegative + 1
                trueNeg.append(id)
            else:
                print 'false negative', str(falseNegative),
'predicted score: ' , str(scorePred), 'Manual score: ' ,
str(scoreMan)
                falseNegative = falseNegative + 1
                falseNeg.append(id)
        if str(scorePred) == str(1):
            if str(scorePred) == str(scoreMan):
                print 'true positive', str(truePositive), 'predicted
score: ' , str(scorePred), 'Manual score: ' , str(scoreMan)
                truePositive = truePositive + 1
                truePos.append(id)
```

```
            else:
                print 'false positive', str(falsePositive),
'predicted score: ' , str(scorePred), 'Manual score: ' ,
str(scoreMan)
                falsePositive = falsePositive + 1
                falsePos.append(str(id))
                falsePos.append(id)

    return truePositive, trueNegative, falsePositive, falseNegative

#function to store predicted values in dictionary
#def dictProgramReview(id, score):
#reviewDict.update({str(id) : str(score)})

#Main function
def main():

    readFileContent()
    print '***********after readFileContent********'

    print '*********************************FALSE
POSITIVE**********************'
    for fp in falsePos:
        print 'fp: ', fp

    print '*********************************FALSE
NEGATIVE**********************'
    for fn in falseNeg:
        print 'fn: ' , fn

    print '*********************************TRUE
POSITIVE**********************'
    for tp in truePos:
        print 'tp: ', tp

    print '*********************************TRUE
NEGATIVE**********************'
    for tn in trueNeg:
        print 'tn: ', tn

main()
```