CS 522

ADVANCED DATA MINING

Project Report

"IMPLEMENTATION OF HEDONOMETER APPROACH ON EMAIL DATASET"

SUBMITTED BY :

SACHIN KRISHNA MURTHY

CWID : A20354077

CONTENTS

IMPLEMENTATION OF HEDONOMETER APPROACH ON EMAIL DATA SET

Abstract:

In this project we are determining the change of feelings of people in an organization over a period of time. Initially the email data set of Enron Corporation is preprocessed with the removal of stop words, splitting up of each line into words and converting into lower case. This is followed by identifying the sentimental words in each email and calculating the sentimental score which is accomplished through the incorporation of Vader Sentiment Lexicon dictionary. Thereafter the sentimental score of all the emails for the each month is summed up and is represented graphically which helps us to identify the variation in the feelings of people in an organization over a period of time.

1. Introduction

1.1 Objective :

The main aim of this project is to determine the change of sentiments of people in an organization over time using Hedonometer approach. This is accomplished for the email data set of Enron Corporation.

1.2 Overview :

Hedonometer is a device used to gauge happiness or sadness and Sentimental analysis is the process of determining whether a piece of writing is positive, negative or neutral.

Using Hedonometer approach, sentimental analysis on email dataset will be performed by extracting the bodies from each emails, obtaining sentiment words which can be used to predict the feelings of the sender and computing the sentiment score for each email by summing up the score of the words for the organization. It is possible to obtain the total sentiment score with respect to the timeline with the help of date field in the data set.

Thereafter obtaining the total sentimental score for each month by summing up the sentimental score calculated for each mail of that month helps in determining the change of sentiments of people in the organization over time.

2. Description of Dataset

We are using the email data set of Enron Corporation. Enron Corporation was an American energy, commodities and services based company. Enron shares reach high of 90$ in the month of August 2000 and Enron stock hits 52 week high of 84.87$ in December 2000. But Enron Corporation was in loss in the year 2001 which went bankrupt in December 2001 and thousands of employees were laid off.

Enron email data set was basically collected and prepared by the CALO Project. This data is valuable because it is the only substantial collection of real email that is public. It consists of a total of 0.5M messages which are the email exchanges from about 150 users who were mostly senior management of Enron Corporation and the data set is organized into folders.
The data set which we are using consists of 20 files.
The emails in the data set consists of five fields: docID, segmentNumber, segmentDate, Subject and content, in the below format:

```
docID:3
segmentNumber:1
Segment Date for 3_1 date: : Mon Dec 04 10:54:00 GMT-06:00 2000
Subject Quotes for the CD's
Content: /* Body of the Email */
```

Example:

```
docID:   3
segmentNumber:          1
Segment Date for 3_1 date: Mon Dec 04 10:54:00 GMT-06:00 2000
Subject Quotes for the CD's.
content:This is the original quote for this project and it did not include the
postage. As soon as I have the details from the vendor, I'll forward those to you.Please call me if you have any questions.
docID:   3
segmentNumber:          2
Segment Date for 3_2 date: Mon Oct 30 05:53:00 GMT-06:00 2000
Subject Quotes for the CD's.
content:We have recieved the quotes from the vendors and the details are:
1 . 150 Cd's with Black printing and the plastc jewel cases - $ 886.502 . The distribution cost (minimum for 500 CD's)   - $ 355.00     This will include the padded envelope, the address labels, the packing and deliver to the post office, but the postage is not included. 3 . The time for Coordination, Artwork and the inserts output and trimming - $ 770.00 So the total amount is approx. $2,011.50. The total time would be about 8 days. These are approximate figures only.
If you have any questions you can call Karen Hunter at X56228. or I can be reached at X55706.
```

From the above example, we can notice that for a single docID there are many segment numbers which indicates the number of exchange of conversations on a single mail.Also we can notice that docID is the indication of separation between emails.

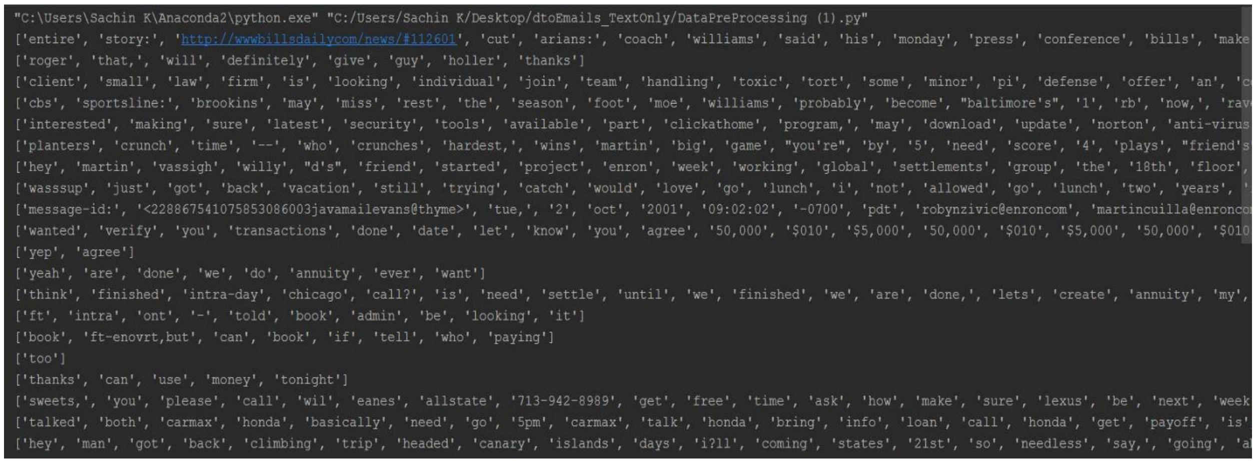
3. Experiments

We have used the approach of "Lexical Analysis" in our experiments.

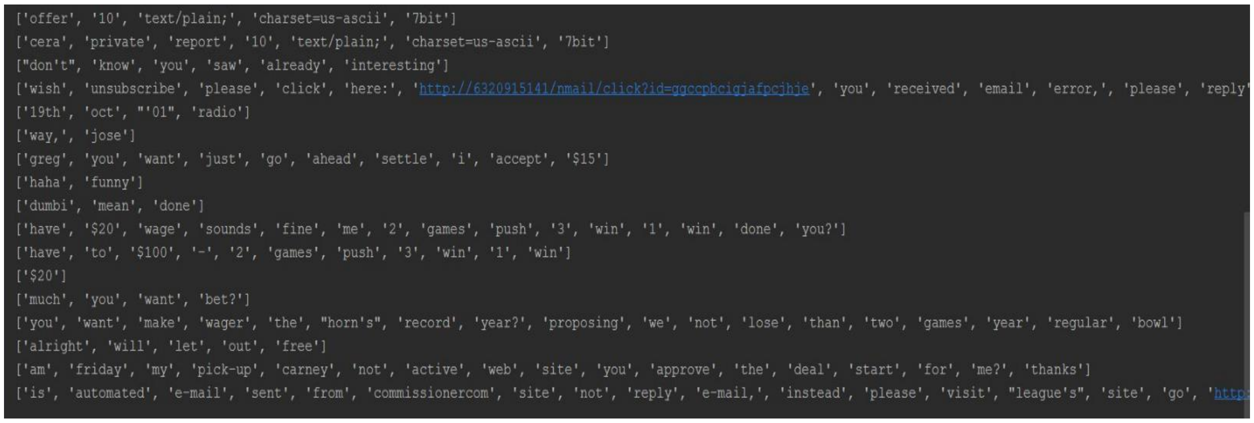
3.1 Data Preprocessing :

- Data preprocessing is the process of removal of irrelevant, redundant, noisy and unreliable data present in the data set.
- Initially each line in the email is split into words and is converted to lower case.
- Then the content and the date field is extracted from each of the email in the data set.
- The data set is cleaned by removal of noise present in the data set.
- Further the stop words such as "a, is, the "etc. present are removed from the content field in the data set.
- Then the preprocessed data is used for obtaining the sentimental words and scores present in each email.

The below screen shots shows the data after preprocessing. Here we can observe the splitting up of words and removal of stop words.



```
"C:\Users\Sachin K\Anaconda2\python.exe" "C:/Users/Sachin K/Desktop/dtoEmails_TextOnly/DataPreProcessing (1).py"
['entire', 'story:', 'http://wwwbillsdailycom/news/#112601', 'cut', 'arians:', 'coach', 'williams', 'said', 'his', 'monday', 'press', 'conference', 'bills', 'make
['roger', 'that,', 'will', 'definitely', 'give', 'guy', 'holler', 'thanks']
['client', 'small', 'law', 'firm', 'is', 'looking', 'individual', 'join', 'team', 'handling', 'toxic', 'tort', 'some', 'minor', 'pi', 'defense', 'offer', 'an', 'c
['cbs', 'sportsline:', 'brookins', 'may', 'miss', 'rest', 'the', 'season', 'foot', 'moe', 'williams', 'probably', 'become', "baltimore's", '1', 'rb', 'now,', 'rav
['interested', 'making', 'sure', 'latest', 'security', 'tools', 'available', 'part', 'clickathome', 'program,', 'may', 'download', 'update', 'norton', 'anti-virus
['planters', 'crunch', 'time', '--', 'who', 'crunches', 'hardest,', 'wins', 'martin', 'big', 'game', "you're", 'by', '5', 'need', 'score', '4', 'plays', "friend's
['hey', 'martin', 'vassigh', 'willy', "d's", 'friend', 'started', 'project', 'enron', 'week', 'working', 'global', 'settlements', 'group', 'the', '18th', 'floor',
['wassup', 'just', 'got', 'back', 'vacation', 'still', 'trying', 'catch', 'would', 'love', 'go', 'lunch', 'i', 'not', 'allowed', 'go', 'lunch', 'two', 'years', '
['message-id:', '<2288675410758530860003javamailevans@thyme>', 'tue,', '2', 'oct', '2001', '09:02:02', '-0700', 'pdt', 'robynzivic@enroncom', 'martincuilla@enronco
['wanted', 'verify', 'you', 'transactions', 'done', 'date', 'let', 'know', 'you', 'agree', '50,000', '$010', '$5,000', '50,000', '$010', '$5,000', '50,000', '$010
['yep', 'agree']
['yeah', 'are', 'done', 'we', 'do', 'annuity', 'ever', 'want']
['think', 'finished', 'intra-day', 'chicago', 'call?', 'is', 'need', 'settle', 'until', 'we', 'finished', 'we', 'are', 'done,', 'lets', 'create', 'annuity', 'my',
['ft', 'intra', 'ont', '-', 'told', 'book', 'admin', 'be', 'looking', 'it']
['book', 'ft-enovrt,but', 'can', 'book', 'if', 'tell', 'who', 'paying']
['too']
['thanks', 'can', 'use', 'money', 'tonight']
['sweets,', 'you', 'please', 'call', 'wil', 'eanes', 'allstate', '713-942-8989', 'get', 'free', 'time', 'ask', 'how', 'make', 'sure', 'lexus', 'be', 'next', 'week
['talked', 'both', 'carmax', 'honda', 'basically', 'need', 'go', '5pm', 'carmax', 'talk', 'honda', 'bring', 'info', 'loan', 'call', 'honda', 'get', 'payoff', 'is'
['hey', 'man', 'got', 'back', 'climbing', 'trip', 'headed', 'canary', 'islands', 'days', 'i?ll', 'coming', 'states', '21st', 'so', 'needless', 'say,', 'going', 'a
```



```
['offer', '10', 'text/plain;', 'charset=us-ascii', '7bit']
['cera', 'private', 'report', '10', 'text/plain;', 'charset=us-ascii', '7bit']
["don't", 'know', 'you', 'saw', 'already', 'interesting']
['wish', 'unsubscribe', 'please', 'click', 'here:', 'http://6320915141/nmail/click?id=ggccpbcigjafpcjhje', 'you', 'received', 'email', 'error,', 'please', 'reply'
['19th', 'oct', "'01", 'radio']
['way,', 'jose']
['greg', 'you', 'want', 'just', 'go', 'ahead', 'settle', 'i', 'accept', '$15']
['haha', 'funny']
['dumbi', 'mean', 'done']
['have', '$20', 'wage', 'sounds', 'fine', 'me', '2', 'games', 'push', '3', 'win', '1', 'win', 'done', 'you?']
['have', 'to', '$100', '-', '2', 'games', 'push', '3', 'win', '1', 'win']
['$20']
['much', 'you', 'want', 'bet?']
['you', 'want', 'make', 'wager', 'the', "horn's", 'record', 'year?', 'proposing', 'we', 'not', 'lose', 'than', 'two', 'games', 'year', 'regular', 'bowl']
['alright', 'will', 'let', 'out', 'free']
['am', 'friday', 'my', 'pick-up', 'carney', 'not', 'active', 'web', 'site', 'you', 'approve', 'the', 'deal', 'start', 'for', 'me?', 'thanks']
['is', 'automated', 'e-mail', 'sent', 'from', 'commissionercom', 'site', 'not', 'reply', 'e-mail,', 'instead', 'please', 'visit', "league's", 'site', 'go', 'http
```

3.2 Usage of Afinn dictionary for the calculation of sentimental score:

Initially we used Afinn dictionary for the calculation of sentimental score. Afinn dictionary consisted of only small number of words and does not provide good coverage from the results obtained and are shown below in the Experimental results section. Thus we incorporated Vader Sentiment Lexicon dictionary.

3.3 Incorporation of Vader Sentiment Lexicon dictionary:

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and a rule based sentiment analysis tool which helps us in understanding the sentiments better. It consists a total of 7517 words and includes:

- List of western style emoticons such as (-: which denotes a smiley and so on.
- Sentiment related acronyms such as LOL which means “Laugh out Loud”.
- Slang words used commonly such as nah, giggly.

VADER sentiment lexicon is sensitive to both polarity and intensity of sentiments expressed.

Vader Sentiment Lexicon consists of features with validated valence scores that indicates both sentiment polarity that is positive or negative and the sentiment intensity on a scale from -4 which is extremely negative to +4 which is extremely positive.

For example, the word "okay" has a positive valence of 0.9, "good" has 1.9, and "great" has 3.1, whereas "horrible" is –2.5 and the frowning emoticon :(is –2.2.

3.3.1 Special Cases Handled:

- To handle punctuations, we have used a list which consists of Punctuation Marks such as “.”, “!”, “?” and so on that enhances the sentiment of the words.
  - For Example, “What an exceptional performance it is!”. In this the exclamation mark increases the sentiment of the sentence.

- To handle negation words, we have used a list which consists of a set of negation words such as “cannot, couldn’t, never” and so on which adds on to the contrary of the sentences which increases the negative sentiment score of the sentence.
  - For example, “He didn’t perform well in the football match”. Here the “didn’t” word adds on to the negative sentimental score of the sentence.

- Also there are some degree modifiers such as intensifiers, booster words which impact the sentiment intensity by either increasing or decreasing it.

  The Booster dictionary consists of set of words such as “amazingly, awfully, exceptionally, remarkably” and so on which boosts the support, force, pressure, effectiveness and so on. These type of words increases the sentiment score both positively and negatively depending on the sentences.
  - For example, “That book is extremely good“. In this case, the word extremely adds more positivity to the sentence which in turn increases the positive sentimental score.
  - Considering the sentence “He hardly does any work”, the word hardly adds more negativity to the sentence.

- Special Case Idioms such as “yeah right”, “the bomb” and so on has been handled by assigning scores for them, so that they will be considered when such type of words appears in the sentence.

- Appropriate measures are taken to remove single letter words such as “a”, “I” and so on. Also the adjacent and redundant punctuation marks has also been removed without affecting the meaning (positivity/negativity) of the sentence.
  - For example, “This is a book”. In this case the single letter word “a” is removed.

3.3.2 Examples :

Case 1: He is smart, handsome and funny.

Sentimental Score : {'neg': 0.0, 'neu': 0.254, 'pos': 0.746, 'compound': 0.8316}

Case 2: He is smart, handsome and funny!

Sentimental Score : {'neg': 0.0, 'neu': 0.248, 'pos': 0.752, 'compound': 0.8439}

Case 3: He is VERY SMART, really handsome, and INCREDIBLY FUNNY!!!

Sentimental Score : {'neg': 0.0, 'neu': 0.294, 'pos': 0.706, 'compound': 0.9469}

Case 4: A really bad, horrible book**.**

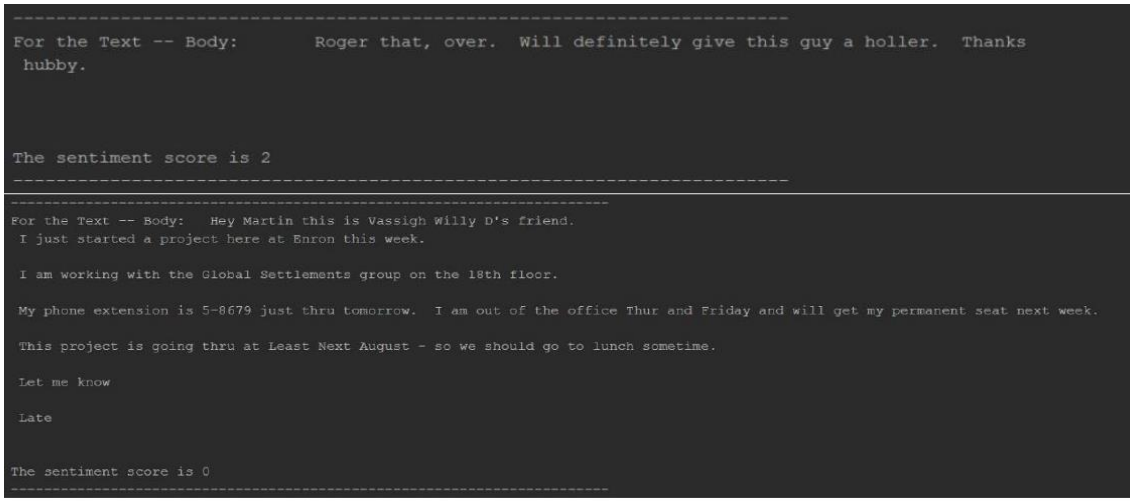
Sentimental Score : {'neg': 0.791, 'neu': 0.209, 'pos': 0.0, 'compound': -0.8211}

- In case 1, the sentence consists of positive words and the sentimental score is calculated as shown above.

- For case 2 the sentence is same as in case 1 with an extra punctuation mark at the end. We could observe the increase in positive score for case 2 when compared with case 1.
- The sentence in case 3 consists of more positive words when compared to case 1 and case 2. Hence we could observe the increase in positive score.
- Case 4 consists of a sentence with negative words, which has increased the negative sentimental score.

4. Experimental Results

4.1 Calculation of Sentimental Score using Afinn dictionary:

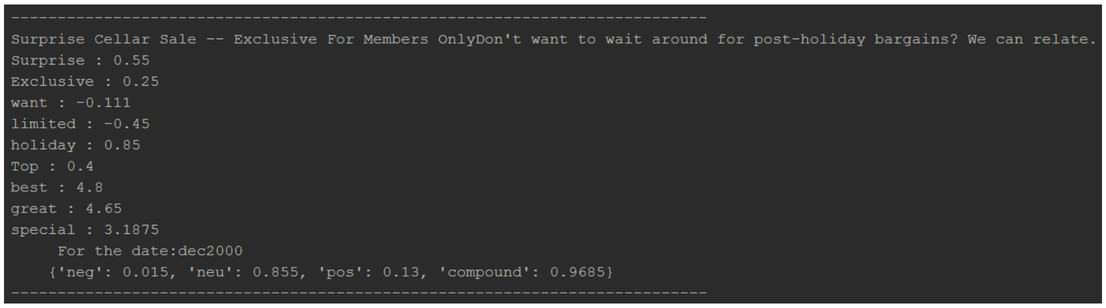


```
--------------------------------------------------------------------------
For the Text -- Body:       Roger that, over.  Will definitely give this guy a holler.  Thanks
 hubby.



The sentiment score is 2
--------------------------------------------------------------------------
```

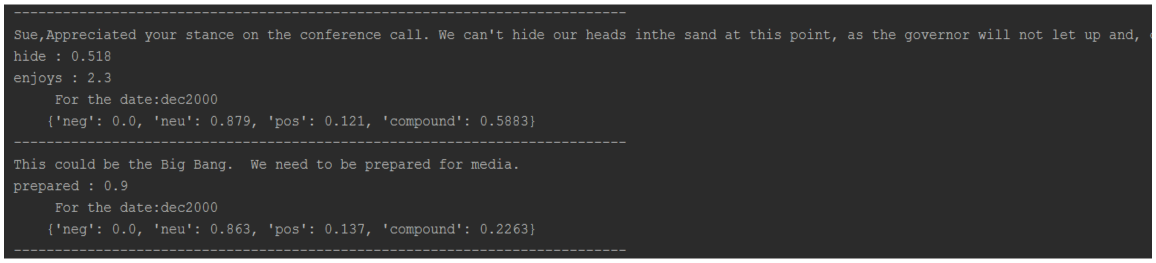
```
--------------------------------------------------------------------------
For the Text -- Body:   Hey Martin this is Vassigh Willy D's friend.
 I just started a project here at Enron this week.

 I am working with the Global Settlements group on the 18th floor.

 My phone extension is 5-8679 just thru tomorrow.  I am out of the office Thur and Friday and will get my permanent seat next week.

 This project is going thru at Least Next August - so we should go to lunch sometime.

 Let me know

 Late


The sentiment score is 0
--------------------------------------------------------------------------
```

From the above screen shots we can notice that, Afinn dictionary was not able to cover all the cases, since the results were not so satisfactory. Hence we decided to use Vader sentiment lexicon dictionary.

4.2 Calculation of Sentimental Score using Vader Sentiment Lexicon Dictionary:



```
--------------------------------------------------------------------------
Surprise Cellar Sale -- Exclusive For Members OnlyDon't want to wait around for post-holiday bargains? We can relate.
Surprise : 0.55
Exclusive : 0.25
want : -0.111
limited : -0.45
holiday : 0.85
Top : 0.4
best : 4.8
great : 4.65
special : 3.1875
     For the date:dec2000
    {'neg': 0.015, 'neu': 0.855, 'pos': 0.13, 'compound': 0.9685}
--------------------------------------------------------------------------
```



```
-------------------------------------------------------------------------
Sue,Appreciated your stance on the conference call. We can't hide our heads inthe sand at this point, as the governor will not let up and,
hide : 0.518
enjoys : 2.3
     For the date:dec2000
    {'neg': 0.0, 'neu': 0.879, 'pos': 0.121, 'compound': 0.5883}
-------------------------------------------------------------------------
This could be the Big Bang.  We need to be prepared for media.
prepared : 0.9
     For the date:dec2000
    {'neg': 0.0, 'neu': 0.863, 'pos': 0.137, 'compound': 0.2263}
-------------------------------------------------------------------------
```

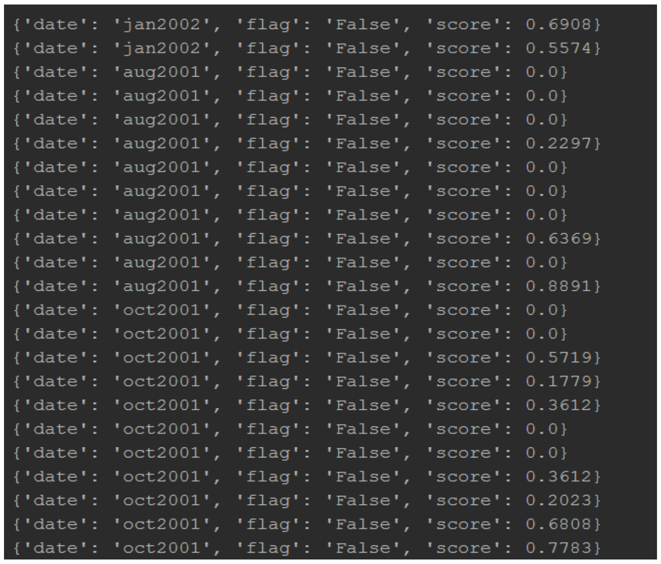
The above screen shots indicates the sentimental scores for each email.

Some of the emails consists of only positive words whereas some of them consists of both positive and negative.

The content of the email is displayed followed by the words which contributes to the sentiment which is positive, negative or neutral along with the scores.

For example, in the first screen shot the word best gives a sentiment score of 4.8 whereas blocked gives a score of -0.45.

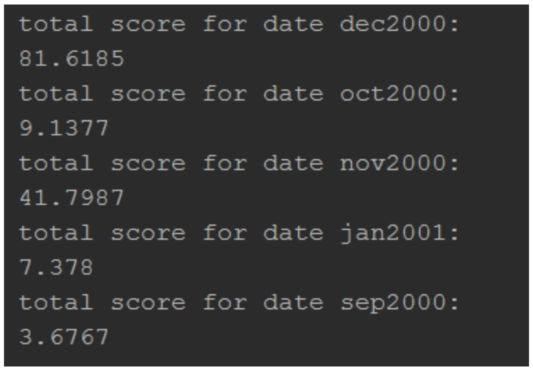
After the calculation of sentimental score of each word, for the particular date, the negative, positive and neutral score is displayed along with the compound score which indicates the total sentimental score of the mail through which we can decide whether the email content is positive or negative. Also the compound is normalized and it will be in the range of -1 which is extremely negative to +1 which is extremely positive.



```
{'date': 'jan2002', 'flag': 'False', 'score': 0.6908}
{'date': 'jan2002', 'flag': 'False', 'score': 0.5574}
{'date': 'aug2001', 'flag': 'False', 'score': 0.0}
{'date': 'aug2001', 'flag': 'False', 'score': 0.0}
{'date': 'aug2001', 'flag': 'False', 'score': 0.0}
{'date': 'aug2001', 'flag': 'False', 'score': 0.2297}
{'date': 'aug2001', 'flag': 'False', 'score': 0.0}
{'date': 'aug2001', 'flag': 'False', 'score': 0.0}
{'date': 'aug2001', 'flag': 'False', 'score': 0.0}
{'date': 'aug2001', 'flag': 'False', 'score': 0.6369}
{'date': 'aug2001', 'flag': 'False', 'score': 0.0}
{'date': 'aug2001', 'flag': 'False', 'score': 0.8891}
{'date': 'oct2001', 'flag': 'False', 'score': 0.0}
{'date': 'oct2001', 'flag': 'False', 'score': 0.0}
{'date': 'oct2001', 'flag': 'False', 'score': 0.5719}
{'date': 'oct2001', 'flag': 'False', 'score': 0.1779}
{'date': 'oct2001', 'flag': 'False', 'score': 0.3612}
{'date': 'oct2001', 'flag': 'False', 'score': 0.0}
{'date': 'oct2001', 'flag': 'False', 'score': 0.0}
{'date': 'oct2001', 'flag': 'False', 'score': 0.3612}
{'date': 'oct2001', 'flag': 'False', 'score': 0.2023}
{'date': 'oct2001', 'flag': 'False', 'score': 0.6808}
{'date': 'oct2001', 'flag': 'False', 'score': 0.7783}
```

After displaying the split up (pos,neg,neu) score for each email, the sentimental score for them with month and year is displayed as shown in the above screen shot.

4.3 Summation of Sentimental Score:



```
total score for date dec2000:
81.6185
total score for date oct2000:
9.1377
total score for date nov2000:
41.7987
total score for date jan2001:
7.378
total score for date sep2000:
3.6767
```

The total sentimental score of all the emails of a particular month is summed up by taking the sentimental score of each email of that month and is displayed as shown above.

For example, the total sentimental score for December 2000 is 81.6185 and the total sentimental score for January 2001 is 7.378 and so on.

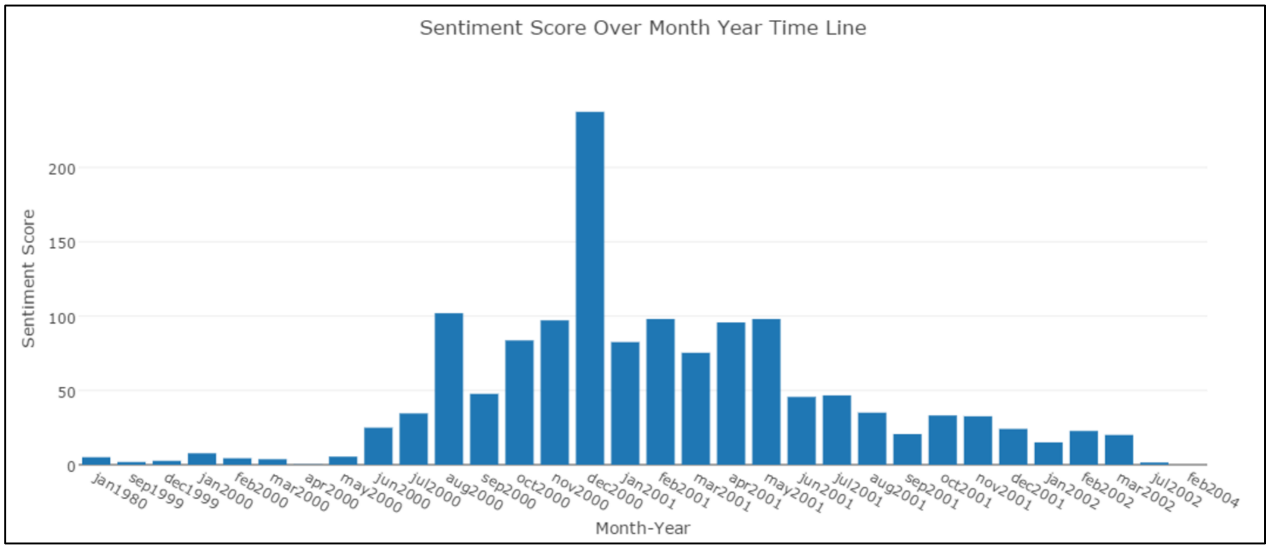
4.4 Comparison using Afinn dictionary vs Vader Sentiment Lexicon dictionary:

Afinn	**Vader Sentiment Lexicon**
A small dictionary with 2477 words.	Consists of 7517 words
Punctuation marks was not handled	Handled
Booster words was not handled	Handled
Special Case idioms not handled	Handled
Normalization of scores was not done	Normalization Done
Single letter words such as ‘a’ , ‘I’ was not removed	Removed

Example,

- When used Afinn, the sentence “no time” was taken as negative. But when used Vader Sentiment Lexicon, “no time” will be taken as neutral.
- Also the word “Please” has been taken as positive depending upon the context of the sentence using Vader Sentiment Lexicon which was not handled using Afinn.

4.5 Graphical Representation:



The above graph gives us the pictorial display of the results of the total sentiment score calculation for each month. The 'x' axis represents the date and the 'y' axis represents the sentiment score.

We can notice how the feelings of the people in an organization changed over the time line. Since the Enron Corporation stock price was high in the month of December 2000, we can say that more number of positive context emails were exchanged in the year 2000. Also we can notice the gradual decrease in the sentimental score from 2000 to 2001, since the company went bankrupt in December 2001.

The sentimental score is high in December 2000 because of the presence of sentimental words in th emails which contribute to the positive majority.

The below table shows the words which most contributed to the sentimental score for December 2000 and January 2002.

December 2000	**January 2002**
good, significant, ensure, best, improve	risk, critical, low, liability, delay

5. Conclusion:

In this project we have analyzed the change of feelings of people in an organization over a timeline by using the Hedonometer approach. Vader Sentiment Lexicon dictionary was used to obtain the sentimental score of each email by predicting the sentiment words in each of them. The sentimental score for each email is calculated depending on the scores of the positive, negative and neutral sentiment words identified in the email. The total sentimental score is obtained by summing up the sentimental scores for each email of the particular month which enables us to decide whether the month has a set of more positive context emails sent or more negative context emails sent. The pictorial representation of the graph enables us to easily identify about the context of the mails sent in a particular month and also the change of feelings of people over a certain timeline.

6. Appendix:

The source code of the project is given below:

```
import os, math, re, sys, fnmatch, string
import plotly.plotly as py
import plotly.graph_objs as go
py.sign_in('cpittapa', 'y3vi46b2iv')
reload(sys)
def make_lex_dict(f):
    return dict(map(lambda (w, m): (w, float(m)), [wmsr.strip().split('\t')[0:2] for wmsr in open(f) ]))
f = 'vader_sentiment_lexicon.txt'
try:
    WORD_VALENCE_DICT = make_lex_dict(f)
except:
    f = os.path.join(os.path.dirname(__file__),'vader_sentiment_lexicon.txt')
    WORD_VALENCE_DICT = make_lex_dict(f)
B_INCR = 0.293
B_DECR = -0.293
c_INCR = 0.733
REGEX_REMOVE_PUNCTUATION = re.compile('[%s]' % re.escape(string.punctuation))
PUNC_LIST = [".", "!", "?", ",", ";", ":", "-", "'", "\"",
             "!!", "!!!", "??", "???", "?!?", "!?!", "?!?!", "!?!?"]
NEGATE = ["aint", "arent", "cannot", "cant", "couldnt", "darent", "didnt", "doesnt",
          "ain't", "aren't", "can't", "couldn't", "daren't", "didn't", "doesn't",
          "dont", "hadnt", "hasnt", "havent", "isnt", "mightnt", "mustnt", "neither",
          "don't", "hadn't", "hasn't", "haven't", "isn't", "mightn't", "mustn't",
          "neednt", "needn't", "never", "none", "nope", "nor", "not", "nothing", "nowhere",
          "oughtnt", "shant", "shouldnt", "uhuh", "wasnt", "werent",
          "oughtn't", "shan't", "shouldn't", "uh-uh", "wasn't", "weren't",
          "without", "wont", "wouldnt", "won't", "wouldn't", "rarely", "seldom", "despite"]
BOOSTER_DICT = {"absolutely": B_INCR, "amazingly": B_INCR, "awfully": B_INCR, "completely": B_INCR, "considerably": B_INCR,
             "decidedly": B_INCR, "deeply": B_INCR, "effing": B_INCR, "enormously": B_INCR,
             "entirely": B_INCR, "especially": B_INCR, "exceptionally": B_INCR, "extremely": B_INCR,
             "fabulously": B_INCR, "flipping": B_INCR, "flippin": B_INCR,
             "fricking": B_INCR, "frickin": B_INCR, "frigging": B_INCR, "friggin": B_INCR, "fully": B_INCR, "fucking": B_INCR,
             "greatly": B_INCR, "hella": B_INCR, "highly": B_INCR, "hugely": B_INCR, "incredibly": B_INCR,
             "intensely": B_INCR, "majorly": B_INCR, "more": B_INCR, "most": B_INCR, "particularly": B_INCR,
             "purely": B_INCR, "quite": B_INCR, "really": B_INCR, "remarkably": B_INCR,
             "so": B_INCR,  "substantially": B_INCR,
             "thoroughly": B_INCR, "totally": B_INCR, "tremendously": B_INCR,
             "uber": B_INCR, "unbelievably": B_INCR, "unusually": B_INCR, "utterly": B_INCR,
             "very": B_INCR,
             "almost": B_DECR, "barely": B_DECR, "hardly": B_DECR, "just enough": B_DECR,
             "kind of": B_DECR, "kinda": B_DECR, "kindof": B_DECR, "kind-of": B_DECR,
             "less": B_DECR, "little": B_DECR, "marginally": B_DECR, "occasionally": B_DECR, "partly": B_DECR,
             "scarcely": B_DECR, "slightly": B_DECR, "somewhat": B_DECR,
             "sort of": B_DECR, "sorta": B_DECR, "sortof": B_DECR, "sort-of": B_DECR}
SPECIAL_CASE_IDIOMS = {"the shit": 3, "the bomb": 3, "bad ass": 1.5, "yeah right": -2,
                 "cut the mustard": 2, "kiss of death": -1.5, "hand to mouth": -2}
def negated(list, nWords=[], includeNT=True):
    nWords.extend(NEGATE)
    for word in nWords:
        if word in list:
            return True
    if includeNT:
        for word in list:
            if "n't" in word:
                return True
    if "least" in list:
        i = list.index("least")
        if i > 0 and list[i-1] != "at":
            return True
    return False

def normalize(score, alpha=15):
    normScore = score/math.sqrt( ((score*score) + alpha) )
    return normScore

def wildCardMatch(patternWithWildcard, listOfStringsToMatchAgainst):
    listOfMatches = fnmatch.filter(listOfStringsToMatchAgainst, patternWithWildcard)
    return listOfMatches

def isALLCAP_differential(wordList):
    countALLCAPS= 0
    for w in wordList:
        if w.isupper():
            countALLCAPS += 1
    cap_differential = len(wordList) - countALLCAPS
    if cap_differential > 0 and cap_differential < len(wordList):
        isDiff = True
    else: isDiff = False
    return isDiff

#check if the preceding words increase, decrease, or negate/nullify the valence
def scalar_inc_dec(word, valence, isCap_diff):
    scalar = 0.0
    word_lower = word.lower()
    if word_lower in BOOSTER_DICT:
        scalar = BOOSTER_DICT[word_lower]
        if valence < 0: scalar *= -1
        if word.isupper() and isCap_diff:
            if valence > 0: scalar += c_INCR
            else:  scalar -= c_INCR
    return scalar

def sentiment(text):
    if not isinstance(text, unicode) and not isinstance(text, str):
        text = str(text)
    wordsAndEmoticons = text.split() #doesn't separate words from adjacent punctuation (keeps emoticons & contractions)
    text_mod = REGEX_REMOVE_PUNCTUATION.sub('', text) # removes punctuation
    wordsOnly = text_mod.split()
    # get rid of empty items or single letter "words" like 'a' and 'I' from wordsOnly
    for word in wordsOnly:
        if len(word) <= 1:
            wordsOnly.remove(word)

    # now remove adjacent & redundant punctuation from [wordsAndEmoticons] while keeping emoticons and contractions
    for word in wordsOnly:
        for p in PUNC_LIST:
            pword = p + word
            x1 = wordsAndEmoticons.count(pword)
            while x1 > 0:
                i = wordsAndEmoticons.index(pword)
                wordsAndEmoticons.remove(pword)
                wordsAndEmoticons.insert(i, word)
            x1 = wordsAndEmoticons.count(pword)
        wordp = word + p
        x2 = wordsAndEmoticons.count(wordp)
        while x2 > 0:
            i = wordsAndEmoticons.index(wordp)
            wordsAndEmoticons.remove(wordp)
            wordsAndEmoticons.insert(i, word)
            x2 = wordsAndEmoticons.count(wordp)

# get rid of residual empty items or single letter "words" like 'a' and 'I' from wordsAndEmoticons
for word in wordsAndEmoticons:
    if len(word) <= 1:
        wordsAndEmoticons.remove(word)
isCap_diff = isALLCAP_differential(wordsAndEmoticons)
sentiments = []
for item in wordsAndEmoticons:
    v = 0
    i = wordsAndEmoticons.index(item)
    if (i < len(wordsAndEmoticons)-1 and item.lower() == "kind" and \
       wordsAndEmoticons[i+1].lower() == "of") or item.lower() in BOOSTER_DICT:
        sentiments.append(v)
        continue
    item_lowercase = item.lower()
    if item_lowercase in WORD_VALENCE_DICT:
        #get the sentiment valence
        v = float(WORD_VALENCE_DICT[item_lowercase])
        if item.isupper() and isCap_diff:
            if v > 0: v += c_INCR
            else: v -= c_INCR
        n_scalar = -0.74
        if i > 0 and wordsAndEmoticons[i-1].lower() not in WORD_VALENCE_DICT:
            s1 = scalar_inc_dec(wordsAndEmoticons[i-1], v,isCap_diff)
            v = v+s1
            if negated([wordsAndEmoticons[i-1]]): v = v*n_scalar
        if i > 1 and wordsAndEmoticons[i-2].lower() not in WORD_VALENCE_DICT:
            s2 = scalar_inc_dec(wordsAndEmoticons[i-2], v,isCap_diff)
            if s2 != 0: s2 = s2*0.95
            v = v+s2
            if wordsAndEmoticons[i-2] == "never" and (wordsAndEmoticons[i-1] == "so" or wordsAndEmoticons[i-1] == "this"):
                v = v*1.5
            elif negated([wordsAndEmoticons[i-2]]): v = v*n_scalar
        if i > 2 and wordsAndEmoticons[i-3].lower() not in WORD_VALENCE_DICT:
            s3 = scalar_inc_dec(wordsAndEmoticons[i-3], v,isCap_diff)
            if s3 != 0: s3 = s3*0.9
            v = v+s3
            if wordsAndEmoticons[i-3] == "never" and \
               (wordsAndEmoticons[i-2] == "so" or wordsAndEmoticons[i-2] == "this") or \
               (wordsAndEmoticons[i-1] == "so" or wordsAndEmoticons[i-1] == "this"):
                v = v*1.25
            elif negated([wordsAndEmoticons[i-3]]): v = v*n_scalar
            onezero = u"{} {}".format(wordsAndEmoticons[i-1], wordsAndEmoticons[i])
            twoonezero = u"{} {} {}".format(wordsAndEmoticons[i-2], wordsAndEmoticons[i-1], wordsAndEmoticons[i])
            twoone = u"{} {}".format(wordsAndEmoticons[i-2], wordsAndEmoticons[i-1])
            threetwoone = u"{} {} {}".format(wordsAndEmoticons[i-3], wordsAndEmoticons[i-2], wordsAndEmoticons[i-1])
            threetwo = u"{} {}".format(wordsAndEmoticons[i-3], wordsAndEmoticons[i-2])
            if onezero in SPECIAL_CASE_IDIOMS:
                v = SPECIAL_CASE_IDIOMS[onezero]
            elif twoonezero in SPECIAL_CASE_IDIOMS:
                v = SPECIAL_CASE_IDIOMS[twoonezero]
            elif twoone in SPECIAL_CASE_IDIOMS:
                v = SPECIAL_CASE_IDIOMS[twoone]
            elif threetwoone in SPECIAL_CASE_IDIOMS:
                v = SPECIAL_CASE_IDIOMS[threetwoone]
            elif threetwo in SPECIAL_CASE_IDIOMS:
                v = SPECIAL_CASE_IDIOMS[threetwo]
            if len(wordsAndEmoticons)-1 > i:
                zeroone = u"{} {}".format(wordsAndEmoticons[i], wordsAndEmoticons[i+1])
                if zeroone in SPECIAL_CASE_IDIOMS:
                    v = SPECIAL_CASE_IDIOMS[zeroone]
            if len(wordsAndEmoticons)-1 > i+1:
                zeroonetwo = u"{} {}".format(wordsAndEmoticons[i], wordsAndEmoticons[i+1], wordsAndEmoticons[i+2])
                if zeroonetwo in SPECIAL_CASE_IDIOMS:
                    v = SPECIAL_CASE_IDIOMS[zeroonetwo]
            if threetwo in BOOSTER_DICT or twoone in BOOSTER_DICT:
                v = v+B_DECR

        # check for negation case using "least"
        if i > 1 and wordsAndEmoticons[i-1].lower() not in WORD_VALENCE_DICT \
            and wordsAndEmoticons[i-1].lower() == "least":
            if (wordsAndEmoticons[i-2].lower() != "at" and wordsAndEmoticons[i-2].lower() != "very"):
                v = v*n_scalar
        elif i > 0 and wordsAndEmoticons[i-1].lower() not in WORD_VALENCE_DICT \
            and wordsAndEmoticons[i-1].lower() == "least":
            v = v*n_scalar
    sentiments.append(v)
# check for modification in sentiment due to contrastive conjunction 'but'
if 'but' in wordsAndEmoticons or 'BUT' in wordsAndEmoticons:
    try: bi = wordsAndEmoticons.index('but')
    except: bi = wordsAndEmoticons.index('BUT')
    for s in sentiments:
        si = sentiments.index(s)
        if si < bi:
            sentiments.pop(si)
            sentiments.insert(si, s*0.5)
        elif si > bi:
            sentiments.pop(si)
            sentiments.insert(si, s*1.5)
for i,sentiment_score in enumerate(sentiments):
    if sentiment_score:
        print wordsAndEmoticons[i] + " : " + str(sentiment_score)
if sentiments:
    sum_s = float(sum(sentiments))
    ep_count = text.count("!")
    if ep_count > 4: ep_count = 4
    ep_amplifier = ep_count*0.292
    if sum_s > 0:  sum_s += ep_amplifier
    elif  sum_s < 0: sum_s -= ep_amplifier
    qm_count = text.count("?")
    qm_amplifier = 0
    if qm_count > 1:
        if qm_count <= 3: qm_amplifier = qm_count*0.18
        else: qm_amplifier = 0.96
        if sum_s > 0:  sum_s += qm_amplifier
        elif  sum_s < 0: sum_s -= qm_amplifier
    compound = normalize(sum_s)
    pos_sum = 0.0
    neg_sum = 0.0
    neu_count = 0
    for sentiment_score in sentiments:
        if sentiment_score > 0:
            pos_sum += (float(sentiment_score) +1) # compensates for neutral words that are counted as 1
        if sentiment_score < 0:
                neg_sum += (float(sentiment_score) -1) # when used with math.fabs(), compensates for neutrals
            if sentiment_score == 0:
                neu_count += 1
        if pos_sum > math.fabs(neg_sum): pos_sum += (ep_amplifier+qm_amplifier)
        elif pos_sum < math.fabs(neg_sum): neg_sum -= (ep_amplifier+qm_amplifier)
        total = pos_sum + math.fabs(neg_sum) + neu_count
        pos = math.fabs(pos_sum / total)
        neg = math.fabs(neg_sum / total)
        neu = math.fabs(neu_count / total)
    else:
        compound = 0.0; pos = 0.0; neg = 0.0; neu = 0.0
    s = {"neg" : round(neg, 3),
         "neu" : round(neu, 3),
         "pos" : round(pos, 3),
         "compound" : round(compound, 4)}
    return s
if __name__ == '__main__':
    from nltk import tokenize
    reachedbody = 0
    para = []
    final=[]
    email=[]
    cnt = 0
    for i in range(1):
        #file_name = "dtoEmails_"+str(i)+"_0.txt"
        file_name = "dtoEmails_1_0.txt"
        datafromfile = open(file_name,"r")

        # for each file, processing each line for body message
        for line in datafromfile:
            line_processed = re.sub('[(){}!.]','',line)
            line_processed = line_processed.lower().split()
            line = line.replace('\n', '')
            if len(line_processed):
                #print(line_processed[0])
                if line_processed[0] == 'docid:':
                    if reachedbody == 1:
                        reachedbody = 0
                        para_str = ''.join(para)
                        #sentences = tokenize.sent_tokenize(para_str)
                        #print "lines_list",lines_list
                        #for sentence in sentences:
                        print para_str
                        ss = sentiment(para_str)
                        if(len(date)>9):
                            d=date[5]+date[9]
                            print "\t For the date:" +d
                        print "\t" + str(ss)
                        email.append({'score': ss['compound'], 'date': d, 'flag': 'False'})
                    para = []
                    date=""
                    cnt=cnt+1
                    print "------------------------------------------------------------------------------"
            if len(line_processed):
                if line_processed[0] == 'segment':
                    date=line_processed[1:]
            if len(line_processed):
                if line_processed[0] == 'content:':
                    reachedbody = 1
                    para.append(line[9:])
                    continue
            if reachedbody == 1:
                para.append(line)
date=[]
for i in range(len(email)):
    print(email[i])
for i in range(len(email)):
    total=email[i]['score']
    f=0
    if ((email[i]['flag'] == 'False') & (len(email[i]['date'])>0) ):
        for j in range(len(email)):
            if(email[i]['date']==email[j]['date']):
                #print tweets[j]
                total=total+email[j]['score']
                email[j]['flag'] = 'True'
                f=1
    if(f==1):
        print 'total score for date '+email[i]['date']+':'
        print total
        final.append({'score': total,'date':email[i]['date']})
        date.append(email[i]['date'])
date=[]
score=[]
import datetime
final.sort(key=lambda final: datetime.datetime.strptime(final['date'], '%b%Y'))
print final
for i in range(len(final)):
    date.append(final[i]['date'])
    score.append(final[i]['score'])
import plotly.plotly as py
data = [
    go.Bar(
        x=date,
        y=score
    ) ]
plot_url = py.plot(data, filename='basic-bar')
```