MOSA V2

Mining Opinion and Sentiment Analysis

Why Do Opinion Matters?

Youtube video comments are very good indicator of the video.It helps us guage peoples reaction to the video.Peoples opinion about the video matters to advertising department of companies.This helps them evaluate the effectiveness of their investment.

Tv Ads comes to youtube before going live(in most cases),therefore advetisers could guage peoples reaction prior to launching ad in TV.

How Can We Guage Opinion?

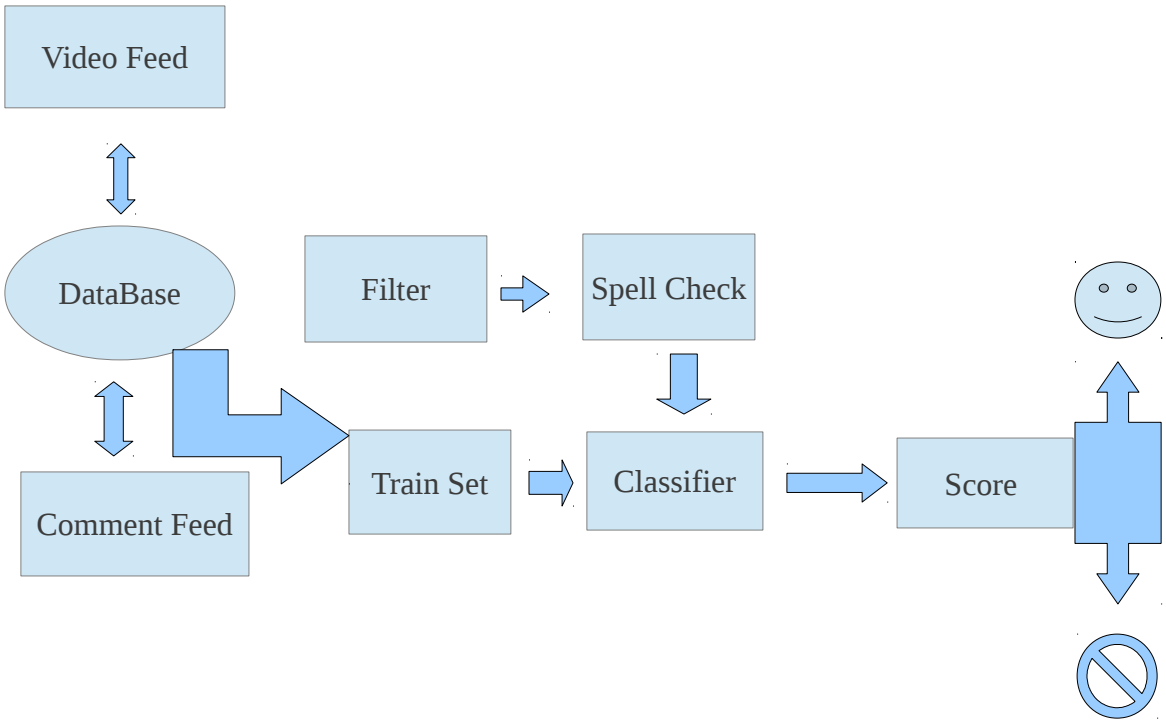
To keep it simple we use + and – approach to opinion classification. By classifying a comment as either positive or negative, we can get a percentage score of positive comment for a video, which is a good estimate of how mass feels about the advertisment.

Whats your approach?

We use machine learning to classify comments. Using training set we train the classifier. As a filtering tool we use tf idf delta score instead of passing on the entire text.

Ref:Opinion mining and sentiment analysis by Pang and lee

This is how it works!



How Do We get Data from youtube?

Using gdata-python-api we extract data from youtube. This can be installed using source file.

```
import gdata.youtube.service
yt_service = gdata.youtube.service.YouTubeService()
query = gdata.youtube.service.YouTubeVideoQuery()
 query.vq = search_terms
 query.orderby = 'relevance'
feed = yt_service.YouTubeQuery(query)
```

The feed we get is in xml format.

Alternatively we can get data is json format by using urllib.urlopen function.

```
url='http://gdata.youtube.com/feeds/api/videos?
&alt=json&q='+SEARCH_TERM+'&start-index='+str(start_index)+'&max-
results=50&v=2&orderby=relevance&uploader=partner&genre=11&paid_content=true&
time=this_month&duration=short&license=youtube&region=IN'
feed=urlopen(url)
feed=json.loads(feed.read().replace('$','s'))
```

Feed that we get is in json format and could be easily inserted in data. There is a minor glitch in inserting the feed directly to mongo db ie '$' could not be at the start of the key. Therefore we replaced it with 's' in entire string.

```
try:
        db.videos.insert(feed);print start_index
except bson.errors.InvalidDocument:
        continue
```

This inserts feed to youtube database and videos collection

Comments can be extracted using the same feed but different api ie

url='https://gdata.youtube.com/feeds/api/videos/'+video_id+'/comments?&alt=json&start-index='+str(start_index)+'&max-results=25'

Video feed api returns 50 video entry in one request, whereas Comment Feed api returns 25 comments. A maximum of 1000 videos and comment could be retreived. Search could be modified by changing url, for specification you can have a look at gdata reference.

In most of the cases there arent 1000 comments for a video, therefore comment feed doesnt cointain entry key in it.

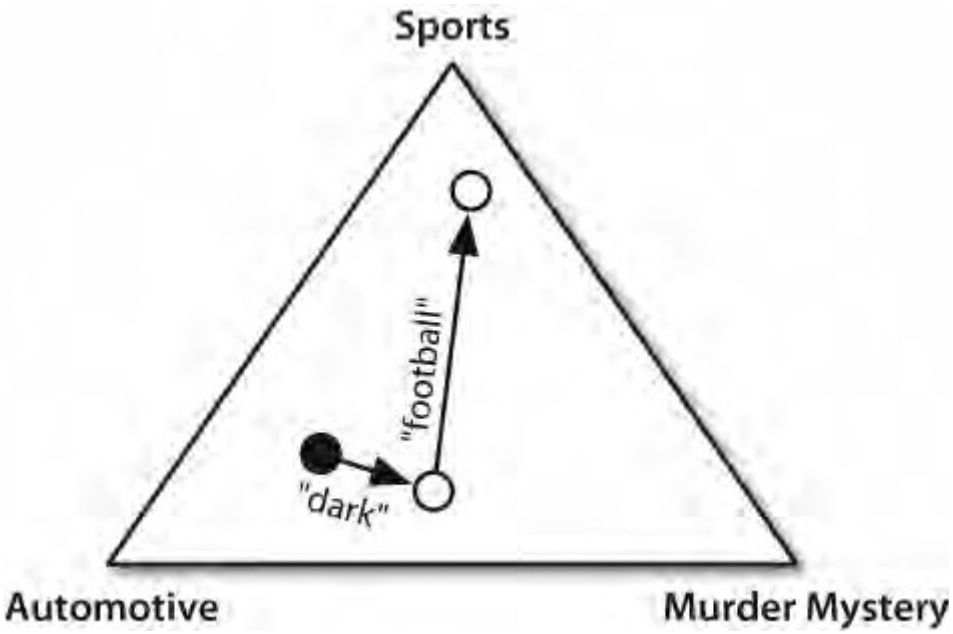
This can be evaluated by studying the feed in python interactive shell.

Based on this internal makeup the mosa feed collector were written.

Ref: python gdata api client
python documnetation for urllib
python documentation for json
python documentation for unicode

Classifier

Naive bayes classifier assigns a probability score to individual class. It assigns conditional probability to each class for a feature. This decides the liklihood of the class for a particular entry.



In this example sports, automotive and murder mystery are classes and dark and football are feature. This way feature help decide class. Feature determines the accuracy of a class. Therefore choice of feature is central to text mining process.

For the purpose of testing i wrote the following code to determine the accuracy of different feature.

#oggpnosn
#hkhr

#creating and testing classifier

```
import ast
from nltk import NaiveBayesClassifier

#-------------------------------------------------------------------------------------------------------------------
from feature_v2 import tf_idf_delta
#-------------------------------------------------------------------------------------------------------------------

import nltk
import random
import pymongo as pym
c=pym.Connection(host='localhost')
db=c['labeled_comment']
comments=db.comment.find()
labeled_comment_tuple=[]
for comment in comments:
        keys=comment.keys()
        index=keys.index('_id')
        if index==1:
                labeled_comment_tuple.append((keys[0],comment[keys[0]]))
        else:
                labeled_comment_tuple.append((keys[1],comment[keys[1]]))

#print labeled_comment_tuple[:100]
#-------------------------------------------------------------------------------------------------------------------
feature_set=[(tf_idf_delta(n),g) for (n,g) in labeled_comment_tuple]
#-------------------------------------------------------------------------------------------------------------------
size=len(feature_set)

random.shuffle(feature_set)
train_set,test_set=feature_set[:int(.85*size)],feature_set[int(.95*size):]
classifier=NaiveBayesClassifier.train(train_set)

print '-------------------------------------------------------------------'
sum=0
for i in range(5):
        sum+=nltk.classify.accuracy(classifier,test_set)*100
accuracy=sum/5
print 'Accuracy:'+str(accuracy)

print'-------------------------------------------------------------------'
classifier.show_most_informative_features(5)
```

You can test your feature by storing it in feature_v2.

NOTE:
Training set composed of
Product Review(14) , Movie review and Manually labelled data(2,000). As of now there are 18,183 labeled comments.

Ref:Cornell Movie Review

UCI Machine Learning Repository

Feature accuracy result of my analysis over train set was:

Feature	Accuracy
Polarity_score(using 5 point scale)	65
Polarity_score(using 2 point scale)	63
Term Presence	64
Term Presence(len(word)>4)	70
Term Presence(len(word)>5)	67
Term Presence(len(word)>6)	69
Term Presence(len(word)>7)	67
Term Presence(len(word)>8)	65
Term Presence(len(word)>9)	65
Term Presence(len(word)>10)	61
Term Frequency(len(word)>1)	69
Term Frequency(len(word)>2)	-

Term Frequency(len(word)>4)	71
Term Frequency(len(word)>5)	67
Term Frequency(len(word)>6)	68
Term Frequency(len(word)>7)	66
Term Frequency(len(word)>8)	66
Bigram	67
Tf-Idf-Delta	74

TF IDF DELTA

This is tf idf delta implementation that i used.

```
all_comment=''
for comment in labeled_comment:
        all_comment+=comment[0]
        all_comment+=' '

all_comment=all_comment.split()
all_comment=set(all_comment)
all_comment=list(all_comment)

#finding total no of positive and negative document



idf={}

def count_comment(labeled_comment,category):
        count=0
        for comment in labeled_comment:
                if comment[1]==category:
                        count+=1
        return count

P=count_comment(labeled_comment,'positive')
N=count_comment(labeled_comment,'negative')
```

```
#making dictionary mapping idf to word


for word in all_comment:
        pt=0;nt=0;import math
        for comment in labeled_comment:
                if word in comment[0]:
                        if comment[1]=='positive':
                                pt+=1
                        elif comment[1]=='negative':
                                nt+=1
        if nt==0:nt=1
        score=float(nt)*P
        if pt==0:pt=1
        score/=((pt)*N)
        score=math.log(score)
        idf[word]=score
idf_sorted=sorted(idf,key=idf.get)


def tf_idf_delta(comment):
        words=comment.split()
        word_count={};score=0
        for word in words:
                if word in word_count:
                        word_count[word]+=1
                else:
                        word_count[word]=1
        for word in words:
                if word not in idf:idf[word]=0
                score+=word_count[word]*idf[word]
        score=round(score)
        return {'tf-idf-delta-score':score}
```

With increased training set it was seen that tf idf delta was gaining accuracy, last accuracy test result was 81%.

POSSIBLE future improvement could be to augment part of speech tagging using nltk tagger, we were able to get 91% accuracy in tagging a sentence(tested against brown corpus).

Following code is implementation of parts of speech tagging to tf idf

delta which failed badly due to my incompetency(accuracy=64%). You can improve that!

```
import nltk
label_tag_words=[]
for label_comment in labeled_comment:
        tag_words=t2.tag(nltk.word_tokenize(label_comment[0]))
        for tag_word in tag_words:
                label_tag_words.append((tag_word,label_comment[1]))

wtl_fdist=nltk.FreqDist(label_tag_words)



def count_comment(labeled_comment,category):
        count=0
        for comment in labeled_comment:
                if comment[1]==category:
                        count+=1
        return count

P=count_comment(labeled_comment,'positive')
N=count_comment(labeled_comment,'negative')

def tf_idf_delta_adv(comment):
        pos_words=t2.tag(nltk.word_tokenize(comment));feature={};score=0;import math
        word_count={};score=0;words=nltk.word_tokenize(comment)
        for word in words:
                if word in word_count:
                        word_count[word]+=1
                else:
                        word_count[word]=1
        for pos_word in pos_words:
                search_this_positive=(pos_word,'positive')
                search_this_negative=(pos_word,'negative')
                if search_this_positive in wtl_fdist:
                        pt=wtl_fdist[search_this_positive]
                else:
                        pt=1
                if search_this_negative in wtl_fdist:
                        nt=wtl_fdist[search_this_negative]
                else:
                        nt=1
                word_score=float(pt*N)
```

```
                word_score/=nt*P;
                word_score=math.log(word_score)
                word_score*=word_count[pos_word[0]]
                score+=word_score
        feature['tf_idf_delta_adv']=score
        return feature
```

You can contribute to accuracy by increasing the training set.
It cointains data in the format.

```
{
        "_id" : ObjectId("51cbec009b10431a2888c667"),
        "comment" : "the subscription service lets me download as much as i want and the player lets me take it wherever i go     ",
        "comment_class" : "positive"
}
```

The following are sample case where tf idf delta fails.Can You Build ON It?

```
How much for another night haahahaha lol
TF IDF DELTA SCORE: {'tf-idf-delta-score': 1.0}
My Prediction: positive
Actual Class: negative
---------------------------------
absolutely not
TF IDF DELTA SCORE: {'tf-idf-delta-score': 1.0}
My Prediction: positive
Actual Class: negative
---------------------------------
it's banned but they still got more than 1million views on this vid alone people  you've been advertised xD
TF IDF DELTA SCORE: {'tf-idf-delta-score': 1.0}
My Prediction: positive
Actual Class: negative
---------------------------------
Aaaawch charlie that hurts 206 LoL
TF IDF DELTA SCORE: {'tf-idf-delta-score': 2.0}
My Prediction: negative
Actual Class: positive
---------------------------------
The problem with dorks is that even their fantasies are dorky
TF IDF DELTA SCORE: {'tf-idf-delta-score': -1.0}
My Prediction: positive
```

Actual Class: negative
---
guess you never heard of EVO2k yea go google that
TF IDF DELTA SCORE: {'tf-idf-delta-score': -0.0}
My Prediction: positive
Actual Class: negative
---
you dont have money your name indicates it geek
TF IDF DELTA SCORE: {'tf-idf-delta-score': 1.0}
My Prediction: positive
Actual Class: negative
---
i cant stop loughing after watching this
TF IDF DELTA SCORE: {'tf-idf-delta-score': 2.0}
My Prediction: negative
Actual Class: positive

We need these types of advertisements not like the boring ones out there Great job KitKat currently having a KitKat waiting to see what happens
TF IDF DELTA SCORE: {'tf-idf-delta-score': 2.0}
My Prediction: negative
Actual Class: positive

Wowhow did they make this vid How did they make the kids dance this way Awesomeness
TF IDF DELTA SCORE: {'tf-idf-delta-score': 3.0}
My Prediction: negative
Actual Class: positive

 Yo Fuckface
TF IDF DELTA SCORE: {'tf-idf-delta-score': 0.0}
My Prediction: positive
Actual Class: negative
---
this ad is so creepy
TF IDF DELTA SCORE: {'tf-idf-delta-score': -0.0}
My Prediction: positive
Actual Class: negative
---
looks like some martian left their kids on earth spy alert
TF IDF DELTA SCORE: {'tf-idf-delta-score': 2.0}
My Prediction: negative
Actual Class: positive

Andaman Islands truth be told though there is a bit of cgi enhancement on the diving shots Andamans does have fantastic diving but not on the day we went
TF IDF DELTA SCORE: {'tf-idf-delta-score': 3.0}
My Prediction: negative
Actual Class: positive

Haha Relax cayetanoluis2 Its pretty clear from your tone that you've got some anger issues and a nasty temper I mean you're almost killing people here and cracking skulls there and calling people who are proud of the country they come from 'cunts' I guess its better if you do India a favor and stay home You don't have the patience tolerance or the understanding of the human condition just yet Its okay not everyone does better luck next time around
TF IDF DELTA SCORE: {'tf-idf-delta-score': -4.0}
My Prediction: positive
Actual Class: negative
---
that looks nothing like the india i live in
TF IDF DELTA SCORE: {'tf-idf-delta-score': -2.0}
My Prediction: positive
Actual Class: negative

They need some decent players
TF IDF DELTA SCORE: {'tf-idf-delta-score': -2.0}
My Prediction: positive
Actual Class: negative
---
lol i didnt expect nike england to show any other sport for nike except football
TF IDF DELTA SCORE: {'tf-idf-delta-score': 3.0}
My Prediction: negative
Actual Class: positive

Greatest ad only INDIANs can understand the thrill in this ad as we have grown playing street cricket great uploadthanx
TF IDF DELTA SCORE: {'tf-idf-delta-score': 2.0}
My Prediction: negative
Actual Class: positive
---
when i grow up i want to be sponsered by nike to use their stuff not the other way around
TF IDF DELTA SCORE: {'tf-idf-delta-score': 2.0}
My Prediction: negative
Actual Class: positive
---
i know shutup
TF IDF DELTA SCORE: {'tf-idf-delta-score': -0.0}
My Prediction: positive
Actual Class: negative

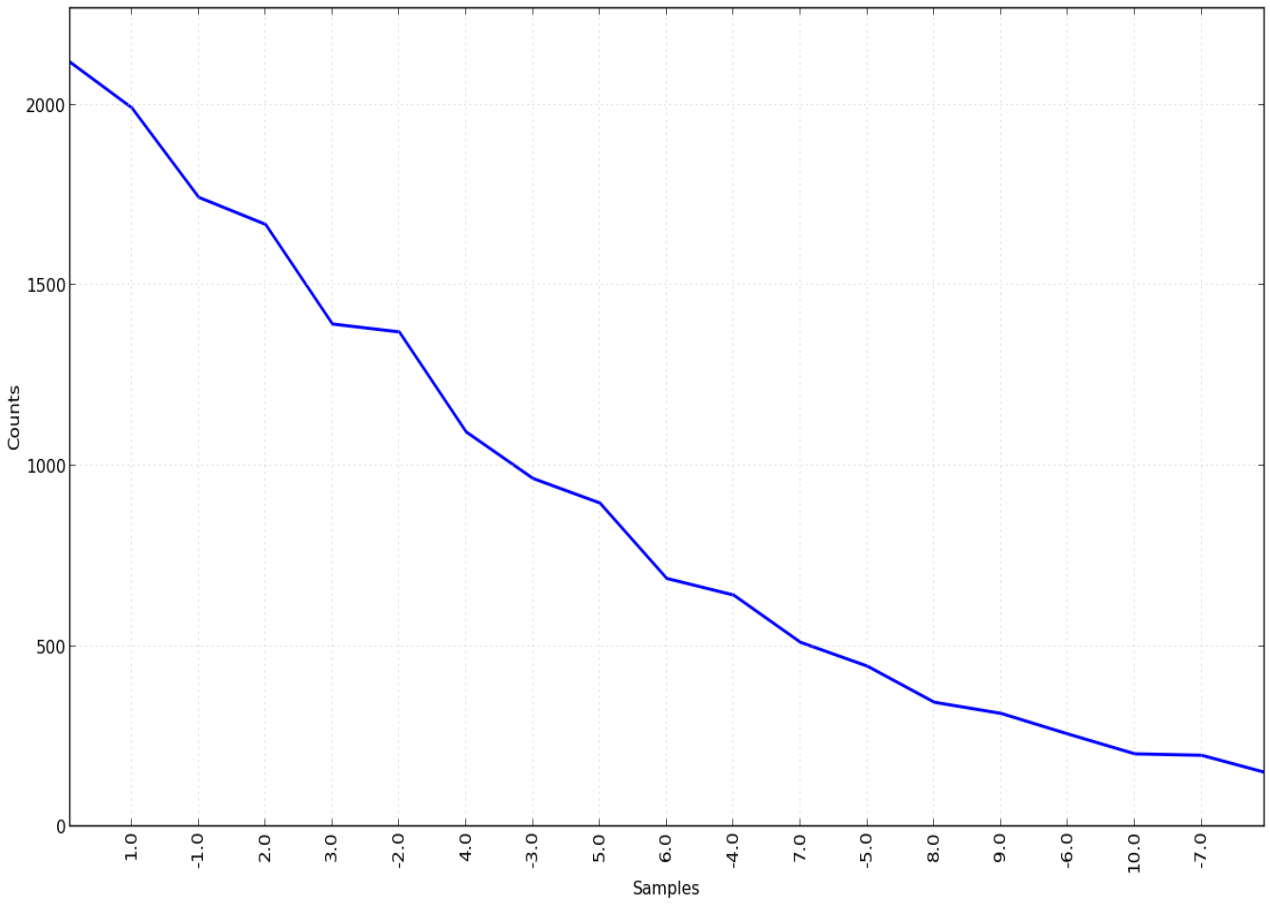
Lebron is great so is Lebron VII shoes it is so comfortabl to wear in the courtI have bought one on the netshare with you the site zoomkole   com
TF IDF DELTA SCORE: {'tf-idf-delta-score': 4.0}
My Prediction: negative
Actual Class: positive

Reminds me of 45 years ago when I was in boarding school After hours we would climb up to the ceiling

and play cricket with the guys in the building next to ours
TF IDF DELTA SCORE: {'tf-idf-delta-score': 2.0}
My Prediction: negative
Actual Class: positive

Baseball has how many people watching it Do you have any forks in your family tree
TF IDF DELTA SCORE: {'tf-idf-delta-score': 1.0}
My Prediction: positive
Actual Class: negative

Frequency distribution for tf idf delta scores was:



Ref: TF IDF DELTA orignal paper
NLTK BOOK

Spell Check

Spell check was implemented in response to failiure cases of tf idf delta feature. People make spelling mistakes very often.

To implement it i took a corpus of misspell word and converted it to python dictionary mapping incorrect with correct word.

```
db=c['misspell_words'] #opening mispell word dictionary stored in database
cursor=db.words.find() #opens cursor to the document
misspell=cursor[0] #stores the dictionary which maps incorrect word to correct word


def spell_check(words): #function to check the words and replace incorrect word with a
correct one
        for word in words:   #goes through all the words
                if word in misspell: #checks whether the word is correctly spelled
                        words.remove(word) #removes incorrect word
                        words.append(misspell[word]) # with correct one :)
        return words #returns the corrected words set back
```

Ref:Stanford NLP coursera course

Misspell 36,000 words data corpus

Filter

```
#oggpnosn
#hkhr

#text preprocessing filters


def filter_it(text):
        text=text.replace('(','')
        text=text.replace(')','')
        text=text.replace(',','')
        text=text.replace('"','')
        text=text.replace('{','')
        text=text.replace('}','')
        text=text.replace('!','')
        text=text.replace('@','')
        text=text.replace('$','')
        text=text.replace('%','')
        text=text.replace('^','')
        text=text.replace('&','')
        text=text.replace('*','')
        text=text.replace('-','')
        text=text.replace('_','')
        text=text.replace('+','')
        text=text.replace('=','')
        text=text.replace('`','')
        text=text.replace('~','')
        text=text.replace('[','')
        text=text.replace(']','')
        text=text.replace('|','')
#       text=text.replace('\r\n','')
        text=text.replace(':','')
        text=text.replace(';','')
```

```
text=text.replace('<','')
text=text.replace('.','')
text=text.replace('?','')
text=text.replace('/','')
text=text.lower()
return text
```

Ref:Data PreProcessing IITM

Whats MOSA_V2?

Its video feed collector, comment feed collector and comment classifier packed in one.

It takes 1000 video feed. For those 1000 feeds it collects comments. This is stored in database. These comments are take from database and classified. Score for each ad is calculated using tf idf delta. This score determines the positivity in advertisment

```
#oggpnosn
#hkhr

#MOSA_V2

#oggpnosn
#hkhr

#mosa video feed collector

#converts the search term to the format in which it has to be supplied
SEARCH_TERM='tv ads commercial india'
SEARCH_TERM=SEARCH_TERM.replace(' ','+')


#
import json
import pymongo as pym
c=pym.Connection()
db=c['youtube']
import ast
```

```
import bson.errors

from urllib import urlopen
for start_index in range(1,1000,50):
	url='http://gdata.youtube.com/feeds/api/videos?
&alt=json&q='+SEARCH_TERM+'&start-index='+str(start_index)+'&max-
results=50&v=2&orderby=relevance&uploader=partner&genre=11&paid_content=true&
time=this_month&duration=short&license=youtube&region=IN'
	feed=urlopen(url)
	feed=json.loads(feed.read().replace('$','s'))
	try:
		db.videos.insert(feed);print start_index
	except bson.errors.InvalidDocument:
		continue


#oggpnosn
#hkhr


#mosa comment feed collector

#making connection to database to retreive video feeds data
import pymongo as pym
c=pym.Connection(host='localhost')
db=c['youtube']


#making cursor to the get access to individual document
cur=db.videos.find()



#importing urllib to deal with url and json to load the data returned by gdata
from urllib import urlopen
import json




#going through all the document in database,getting video id, which is used to get
comments
for entry in cur:
	for video in entry['feed']['entry']:
		video_id=video['mediasgroup']['ytsvideoid']['st']
```

```
                print video['mediasgroup']['mediastitle']['st']
                for start_index in range(1,1000,25):                #to get all the comment

        url='https://gdata.youtube.com/feeds/api/videos/'+video_id+'/comments?
&alt=json&start-index='+str(start_index)+'&amp;max-results=25'
                        feed=urlopen(url)  #fetching data returned by url

                        feed_text=feed.read().replace('$','s')              #cant store $ as
starting word in key
                        try:
                                feed=json.loads(feed_text)          #converts text to json or
dictionary
                                if 'entry' in feed['feed']:      #to remove all the entries that do not
cointan
                                        db.comments.insert(feed)
                        except ValueError:
                                continue


#oggpnosn
#hkhr

# MOSA V2

import nltk
import math
#getting data to configure training set for the classifier

import pymongo as pym #importing library pymongo to get essential
c=pym.Connection(host='localhost') #Making Connection to database train_set
db=c['train_set'] #database handle top train set
comments=db.comment.find() # Cursor to train set
label_comment=[] #list in which training set will be stored as collection of tuple in the
format [(comment,comment_label)....]

for comment in comments:  #acess to individual document in train_set
        label_comment.append((comment['comment'],comment['comment_class']))
#appends data to label_class a list of tuple

#fetching data from misspell_words to implement spell checker


db=c['misspell_words'] #opening mispell word dictionary stored in database
cursor=db.words.find() #opens cursor to the document
misspell=cursor[0] #stores the dictionary which maps incorrect word to correct word
```

```
def spell_check(words): #function to check the words and replace incorrect word with a
correct one
        for word in words:   #goes through all the words
                if word in misspell: #checks whether the word is correctly spelled
                        words.remove(word) #removes incorrect word
                        words.append(misspell[word]) # with correct one :)
        return words #returns the corrected words set back


#getting fdist of words in trainset
P=0
N=0
word_label=[]

for comment in label_comment:
        for word in comment[0].split():
                word_label.append((word,comment[1]))
        if comment[1]=='positive':
                P+=1
        else:
                N+=1

fdist=nltk.FreqDist(word_label)



#tf idf delta feature

def tf_idf_delta(comment):
        words=comment.split()
        words=spell_check(words)
        word_count={}
        su=0
        count=0
        for word in words:
                if word in word_count:
                        word_count[word]+=1
                else:
                        word_count[word]=1
        for word in words:
                pt=fdist[(word,'positive')]
                nt=fdist[(word,'negative')]
                if pt==0:pt=1
```

```
                if nt==0:nt=1
                score=float(pt)/nt
                score*=N
                score/=P
                score=math.log(score)
                score*=word_count[word]
                su+=score;
        su=round(su)
        return {'tf_idf_score':su}



#preparing Training set

train_set=[(tf_idf_delta(n),g) for (n,g) in label_comment]
classifier=nltk.NaiveBayesClassifier.train(train_set)

#getting comments from youtube


db=c['youtube']
from datetime import datetime
cursor=db.comments.find()
cursor_length=cursor.count()

#get_title() to fetch title for a given video_id

def get_title(video_id):
        cur=db.videos.find()
        for videos in cur:
                if 'entry' in videos['feed']:
                        for i in range(0,50):
                                try:
                                        if video_id==videos['feed']['entry'][i]['mediasgroup']
['ytsvideoid']['st']:
                                                return videos['feed']['entry'][i]['mediasgroup']
['mediastitle']['st']
                                except IndexError:
                                        break



for start_index in range(0,cursor_length,40):
        total_result=cursor[start_index]['feed']['openSearchstotalResults']['st'];print
'----------';print total_result
```

```
        cycles=total_result/25      #no of cycles of 25 comment to be executed
        if cycles==0:cycles=1;      #if less than 25 comments
        if cycles>40:cycles=40      #if cycles are greater than 40
        to_be_inserted={};ce='terminal'
        if 'entry'  in cursor[start_index]['feed']:                    #to avoid cases in which there
arent any comment
                video_id=cursor[start_index]['feed']['entry'][0]['ytsvideoid']['st'];
                title=get_title(video_id);print title
                score=0;count=0
                for index in range(start_index,start_index+cycles):     #going through all
cycle
                        if 'entry'  in cursor[index]['feed']:
                                for i in range(0,49):
                                        try:
                                                if cursor[index]['feed']['entry'][i]['ytsvideoid']['st']!
=video_id:ce='exit';break
                                                comment=cursor[index]['feed']['entry'][i]
['content']['st'];
                                                if title=="How I Met Your Mother - Ted's
House":print comment

        comment_class=classifier.classify(tf_idf_delta(comment));
                                                if comment_class=='positive':
                                                        score+=1
                                                count+=1
                                        except IndexError or KeyError:
                                                break
                                if ce=='exit':break
                score=float(score)/count;
                score*=100;print score;print count
                to_be_inserted['title']=title
                to_be_inserted['score']=score;
                to_be_inserted['time']=datetime.now().isoformat()
                to_be_inserted['video_id']=video_id
                to_be_inserted['comments_evaluated']=count
                db.results.insert(to_be_inserted);print '------------------------'
```

>By changing the search words one can get feed for non advertisment.
>By increasing train set accuracy could be increased
>By modifying algorithm or using pos tagging using bigram tagger one can make use of context

Ref:Codes for MOSA_V2

How Can I Contact You?

You can mail your queries at tanaygahlot@gmail.com