What Tweets Could Say About Movies

Final Project Report

Group A8

1. Project Introduction:

The 'Movie' dataset includes almost all the records on Twitter with hashtags about 28 movies that were released in 2017 and 2018. Our target is to extract useful information from those tweets and try to use the information to help the decision-making for the film producers or the entertainment company.

To handle the big data, we use the virtual box, python, excel, R, and Tableau. We unfold the nested data and clean the invalid data. To use sentiment analysis, we deal with the text and standardize each attribute.

Following are the essential facts about our findings and opinions:

- People in California would tweet more than those in other states.
- The sentiment is somehow affected by location.
- The box office is related to the heat and the overall attitudes on twitter, but exceptions exist.
- Since the box office is related to the sentiment, improving sentiment is important. Location is a crucial factor to be considered for marketing when the movie initially launches. Besides, when deciding the overseas launches, the producer should choose the target country by thinking about the country's taste. For example, Portugal does not present a negative attitude toward comedy and horror.

2. Problem Statement:

Do the retweet count and likes count appear to be related to the total gross revenue of the movies? For people in different countries, do they have identical attitudes towards different genres? Is there a relationship between total gross and sentiment scores? Are there any rules in people's attitude toward movies among different regions?

3. Dataset Description:

The 'Movie' dataset is a JSON file extracted from Tweeter, about 923 MB in size. This dataset includes almost all the records on Twitter with hashtags about 28 movies that were released in 2017 and 2018. This is big data because the volume is huge – it contains more than 200 thousand of records that could not simply be handled by tools like Excel or SQL that we usually use.

Besides, the original data is very unstructured and needs several steps to extract information that is useful to us.

The information could be divided into three main parts: the tweet content, the user's basic information, and the retweet's content if available.

- The tweet's content includes the tweet's id, likes count, reply count, the image URL, and most importantly, the text.
- The user's basic information includes the name, the user's id, description, follower counts, friend counts, also some home page information, such as the image URL and the friends count.
- The retweet's content includes the status of the original tweet, including the likes count, reply count, and image URL.

4. Methods：

Part I Processing

1) JSON to structured data

Initially, we use the Cloudera terminal to get structured data because of the faster handling speed. We unfolded the nested data and extracted the data we want, the basic info about the tweets and some crucial attributes about the users, such as the location, description, and the follower counts. We created a python code in the terminal and ran it to put our wanted results in CSV.

2) Standardizing data

(1) Hashtags

Our goal is to make Hashtags only contain the movie names we need. The tool we used is python. The original column of Hashtags contains 'None' values, so we were trying to find if there are movies name in the Text, so we can extract the names and replace them to "None" values. Second, we strip extra punctuation marks like '[ ]' and ' '. The movie names in the document also written in different ways. For future analysis purposes, we regulated them by lowering the cases.

(2) Location

The 'location' was manually typed in text type by users, which means there were plenty of problems, including different formatting (different cases for one place), different names (e.g. for those who live in Santa Monica, some typed 'Santa Monica', some typed 'Santa Monica, CA', some typed 'Ca'), missing (in fact, nearly half of the users have 'none' in their file), typos, unrelated information, etc. To have structured location information for further use, we have to formalize it into certain levels. In this case, we formalized all the locations into states (for those

inside the US) and countries (for those outside the US) as we want to analyze the tweets in terms of area within and out of the US. We also transformed all US locations into 'USA' for potential use if we need to analyze at the country level.

To achieve this, we used Python with the following steps:

- Create lists of 50 states and territories of the US and lists of countries that appear in the dataset.
- Manually put the possible names of a certain area into the list we created for it. Because there are so many different expressions for the same area, this step requires lots of work.
- Rename each location to state/country name if it matches the value in the lists and put them into a new column.
- Rename each location to 'USA' if it is a state of the US and put them into a new column.

Finally, we get two new columns, one contains US states and other countries, and the other contains countries. We have 110 thousand of records that have valid locations after these steps.

(3) Text

Initially, there are many invalid symbols and emoji in the texts, and also we have many records that are not in English. Thus, we have the following steps to handle the text:

- Clear the emoji and useless symbols: By importing the emoji package in python, we removed the emojis and other unnecessary symbols.
- Detect the language and translate to English: In order to decrease the workload, we first add a column to put the detected language, and then we screen the records which are not written in English. Although we tried goslate and google trans at first, the functions are a little unstable because of the IP blocking problems.

5) Calculation

Sentiment Analysis could bring us a more precise view of the attitudes of the audience. And conducting the analysis about attitudes could provide us with more valuable information.

We use the library Text Blob to get the estimating sentiment scores for every tweet.

Part II Analyze & Visualization

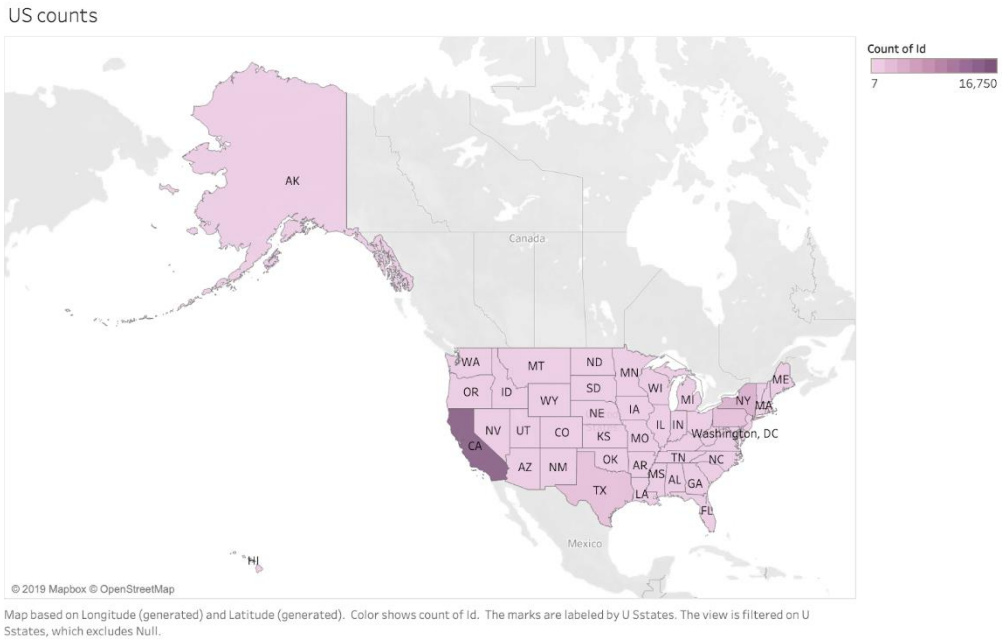
Before visualization, we used the R and boosting tree to find the relationship between sentiment and other attributes.

We use Tableau as our visualization tool. We conduct inner join on the organized datasets we have. The first graph we created is the relationship of tweets' likes and retweets between total gross of movies. We eliminate some movies because there is no data on their box office. We change the measure of total gross to Maximum and Average of likes count and retweet counts. We choose row as movie, columns as average sentiment scores and maximum total gross for the second graph. For graphs in 'Attitudes Towards Genre', we use genre and countries as columns, the average sentiment score as the row to see the attitude of various countries towards distinctive movie genre. For the last graph, we try to find some rules among states in America. We choose latitude as row, longitude as column to get a map. Then we count the number of tweets in each state and use color depth to visualize it.

5. Conclusion:

1) Audience Behavior

Figure 1



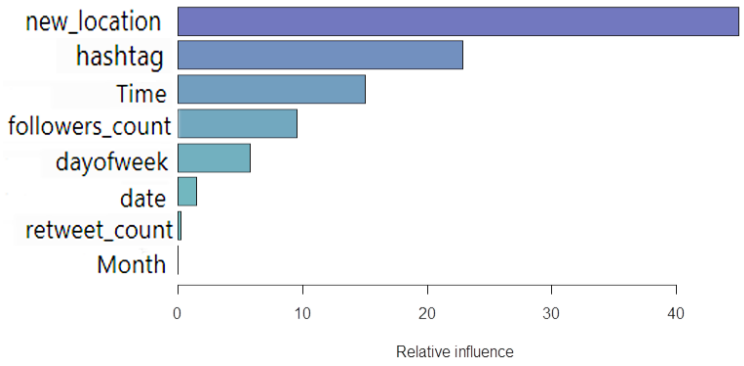
Map based on Longitude (generated) and Latitude (generated). Color shows count of Id. The marks are labeled by U Sstates. The view is filtered on U Sstates, which excludes Null.

From *figure 1*, after applying the color-scale, it could be visually observed that CA is the state where people tweet most about movies. Meanwhile, in this map, Texas and New York also have a darker color than other states. People in these two states also tweets a lot about these movies. As shown by relative statistics, in 2017 and 2018, California, Texas, and New York are three states with the highest total GDP among all states in America. This gives us a hint that people might pay more attention to the movies if they are economically advantageous. This makes sense because people will spend more money on recreation and entertainment when they get rich. Thus, the movie industry can work more on states with higher GDP when promoting their films, like the three states we mentioned above. Targeting at people with a higher probability of paying for movies will save costs and earn more profit for the company.

2) Audience Sentiment

We care about the audiences' feedback about the movie, which would influence the heat. How could sentiment be effected? We run a boosting forest and see the relative influence of each attribute below.

Figure 2

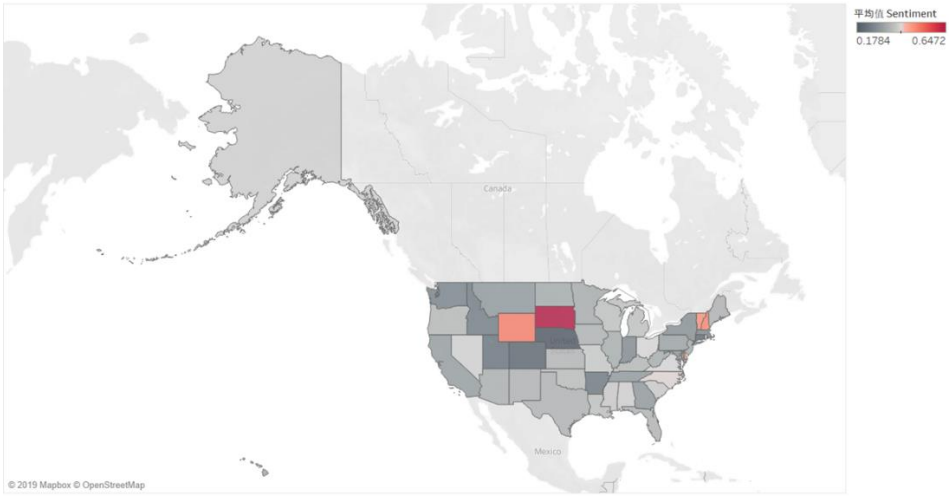


What is very interesting here is that the effect of location is even more apparent than the effect of the movie itself. It seemed that audiences in some places would be more tolerant or more willing to give positive feedback, while someone in other areas would be meaner or more likely to give negative feedback.

Another fun fact is that time, and followers count of tweets would also be somehow important. Thus, we could dive into those attributes to see if there is any valuable information.

As we could see from the following map, people in South Dakota, Wyoming, Vermont, and New Hampshire are more likely to give more positive comments on movies. When doing the premier or marketing activities, the producer could consider those states as their choice.

Figure 3- Attitudes Towards Movie (Different States)



When scoping into the global insight, we also want to know if different countries would hold different attitudes towards distinct genres.

We observe some countries have negative attitudes toward certain genres (See figures in Appendix). People in Portugal dislike Comedy and Horror movies quite a lot, and Kuwait people are not fans of Biography. Production should choose the countries to promote their movies wisely, pinking countries with positive attitudes.

It seems that European and North American countries have higher sentiment scores than other countries in all three kinds of movies we list. Despite this pattern, we also need to consider that there are fewer people use twitter in other continents like Africa and Asia. The sentiment score of countries like Thailand may not be representative. So, it's hard to say whether there's a preference rule among different continents.

3）Relation with box office

Figure 5

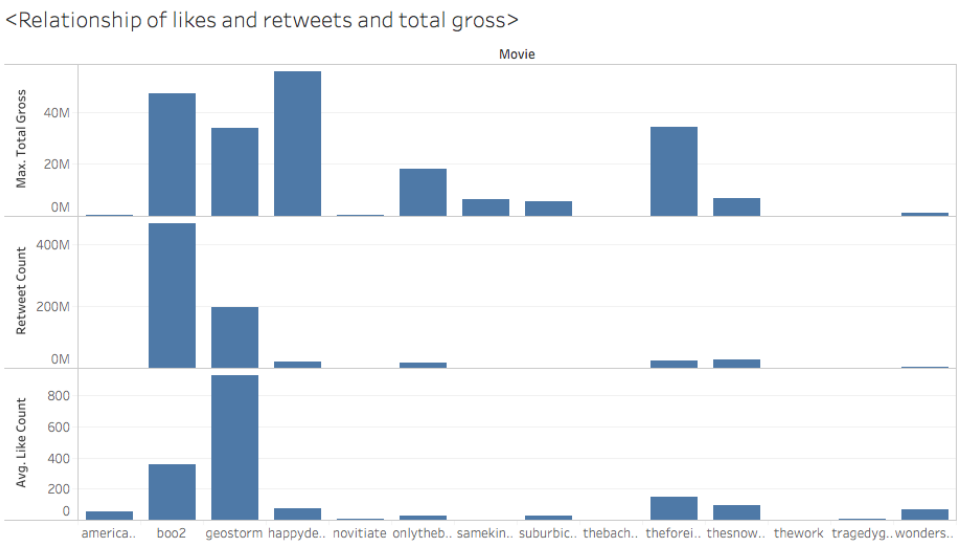
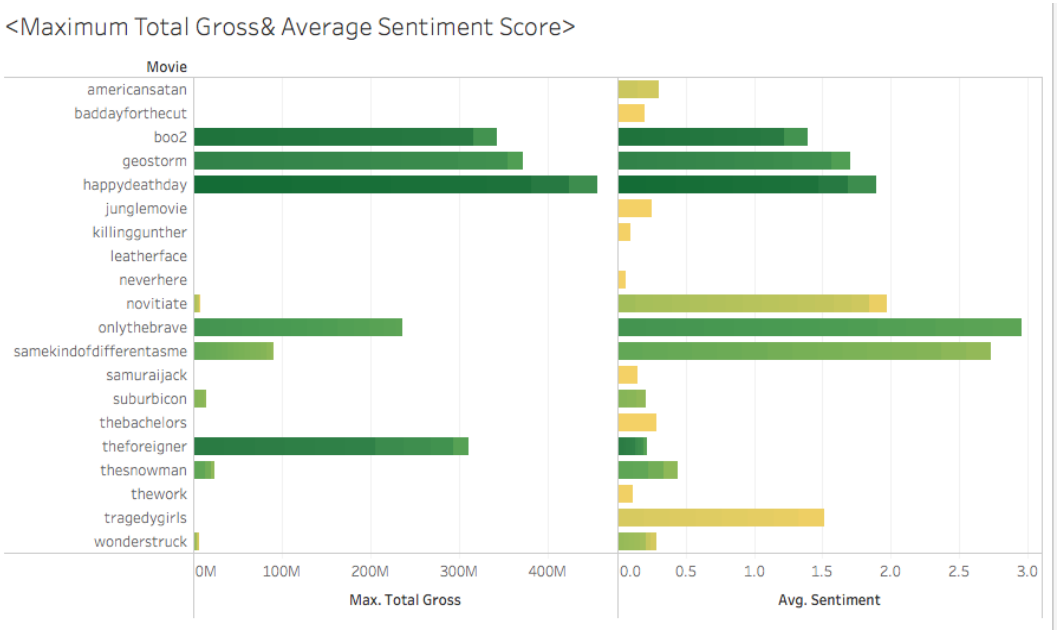


Figure 5 illustrates the relationship between likes and retweets and the maximum total gross of movies. We can say that there is a positive relationship between those three variables; movies with more likes and retweets have a higher box office, like *Boo2* and *Gro storm*. The production company should have promotions on twitter to bring attention from twitter users for the future growth of the box office since there are many movie lovers on this platform. However, there is also a movie like *Happy Death Day* with low likes and retweets but high gross. *Happy Death Day* is the type of movie that went viral after its release, which tells us the promotion and the high expectations are not the determined keys to a high box office, the quality of the movie is important as well.

Figure 6



Overall, positive sentiment scores movies do not have a lower gross in the box office. There are one or two movies like The Foreigner and Tragedy Girls. The Foreigner has a high gross, but a low sentiment score; Tragedy Girls has little gross but relatively high sentiment score. It might because Tragedy Girls only had eight weeks showtimes in theatres. The cumulative gross is lower, but we can say that making high-quality movies that audiences would like could expect the repay of the gross.

Appendix

Figures: Attitudes Towards Genre (Different Countries)

Figure 4.1 - Biography

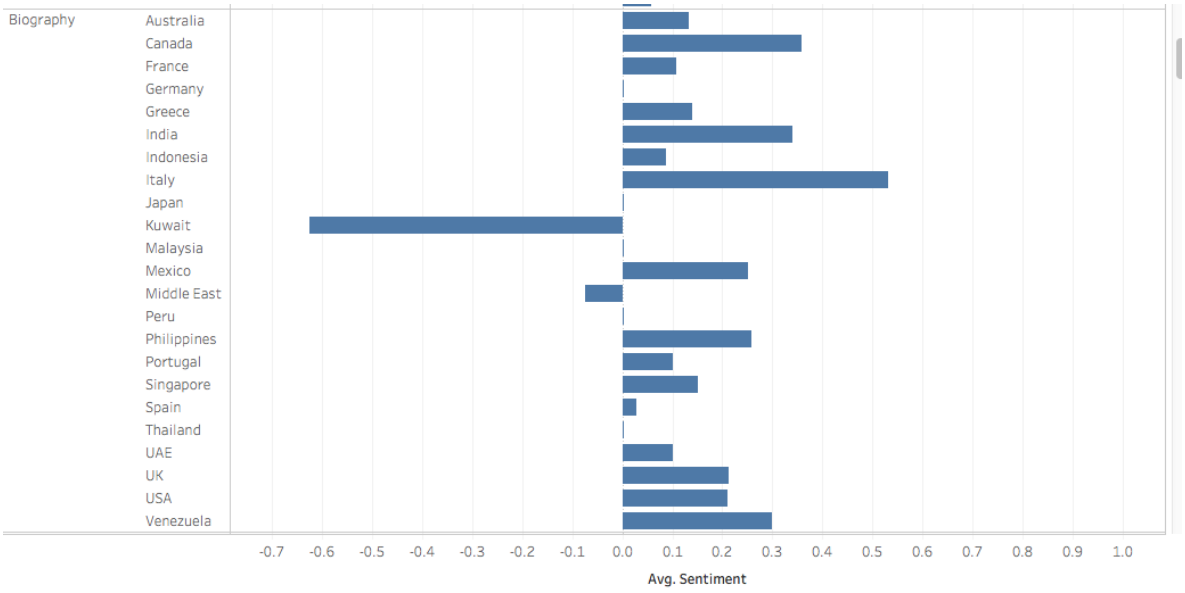


Figure 4.2 - Comedy

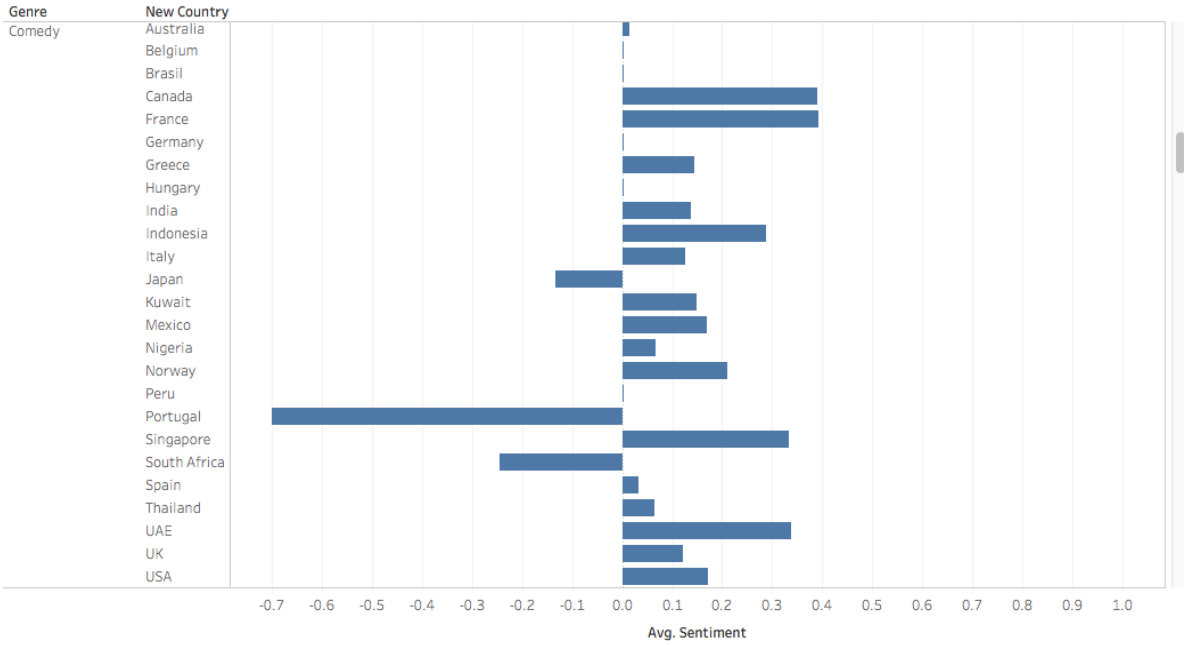
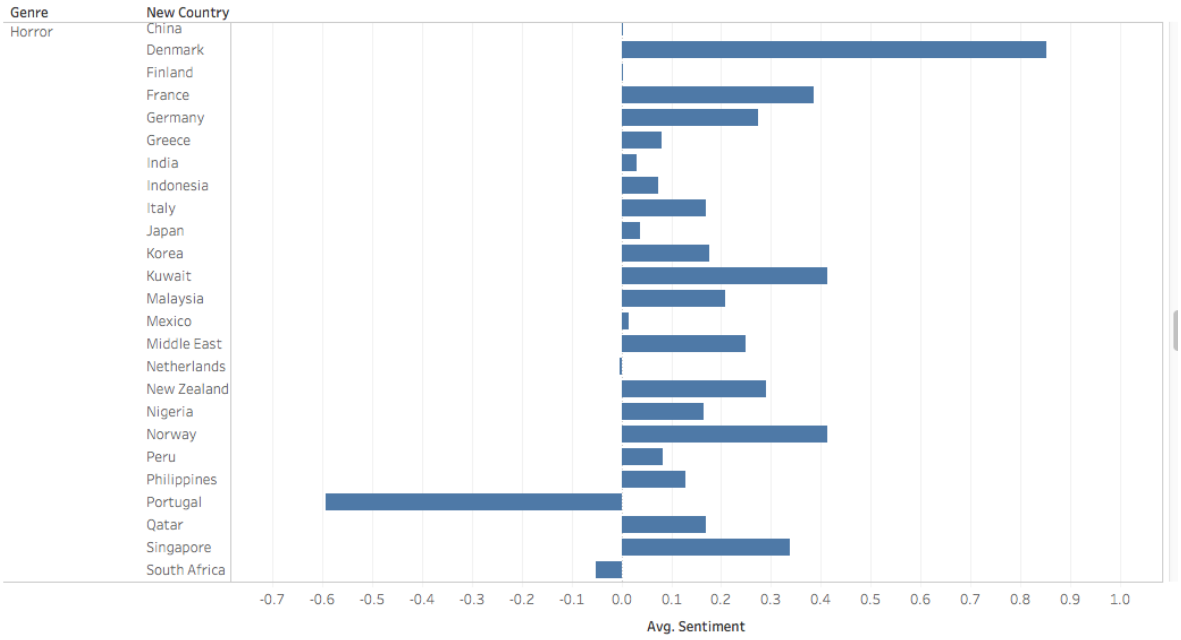


Figure 4.3 - Horror



Coding records:

- Extracting and structuring data from JSON file,
- Text cleaning and translation,
- Hashtags extraction and filtering,
- Location structuring,
- Sentiment analysis.

```
import json
import codecs
import sys
n=0
UTF8Writer = codecs.getwriter('utf8')
sys.stdout = UTF8Writer(sys.stdout)
def extract(a):
	try:
		text=a['text'].replace('\n','')
	except:
		text='None'
	try:
		retweet_count=a['favorite_count']
	except:
		retweet_count='None'

	try:
		like_count=a['retweet_count']
	except:
		like_count='None'
	try:
		reply_count=a['reply_count']
	except:
		reply_count='None'
	try:
		created=a['created_at']
	except:
		created='None'
	try:
		id=a['id']
	except:
		id='None'

	hashtags=[]
	try:
		for one in a['entities']['hashtags']:
			hashtags.append(one['text'].encode('ascii'))
	except:
		hashtags='None'
	hashtags=str(hashtags)
	try:
		followers_count=a['user']['followers_count']
	except:
		followers_count='None'
	try:
		location=a['user']['location']
	except:
		location='None'
	try:
		description=a['user']['description'].strip().replace('\n','')
	except:
		description='None'
	try:
		userid=a['user']['id']
	except:
		userid='None'
	try:
```

```
        username=a['user']['screen_name']
    except:
        username='None'
    try:
        retweet_id=a['retweeted_status']['id_str']
    except:
        retweet_id='None'
    print '[%s],[%s],[%s],[%s],[%s],[%s],[%s],[%s],[%s],[%s],[%s],[%s],[%s]'
%(id,text,retweet_count,reply_count,like_count,created,hashtags,username,userid,followers_count,location,descripti
on,retweet_id)

n=n+1
for line in sys.stdin:
    try:
        a=json.loads(line)
    except:
        n=n+1

    extract(a)
    try:
        b=a['retweeted_status']
        extract(b)
    except:
        n=n+1
```

In [1]:
```
from googletrans import Translator
import pandas as pd
import emoji

df_movies=pd.read_csv("movie_data2.csv",low_memory=False)
df_movies["deal"]=df_movies["text"].apply(lambda x: emoji.demojize(x.replace(
'&','')))
#df_text=pd.read_csv("movie_data_onlytxt.csv")
#df_movies=pd.read_csv('movie_data_2.csv')
```

In [4]:
```
df_text=df_movies[df_movies['retweet_id']=='None'][['Unnamed: 0','id','deal']]
```

In [5]:
```
from langdetect import detect
def define_lan(x):
    lan=''
    try:
        lan=detect(x)
    except:
        lan="can't define"
    return lan

df_text['lan']=df_text['deal'].apply(define_lan)
```

In [6]:
```
df_text.to_csv("movie_data_text.csv",index=False,sep=',',encoding='utf_8_sig')
```

In [35]:
```
from textblob import TextBlob
import goslate
gs = goslate.Goslate(retry_times=6000)
df_text=df_text.drop_duplicates(subset=['id','deal'],keep='first')
#df_text.groupby(by=['lan']).count()
df_needtrans=df_text[df_text['lan']!='en']
n=0
def trans(text):
    translatedText = gs.translate(text[0],'en')
    return str(translatedText)
def trans2(x):
    blob = TextBlob(x[0])
    try:
        translatedText=blob.translate(from_lang=x[1], to='en')
    except:
        translatedText=x[0]
    return str(translatedText)
#df_needtrans['translate']=df_needtrans[['deal','lan']][:100].apply(trans,axis
=1)
```

In [ ]:
```
df_needtrans['translate']=df_needtrans[['deal','lan']].apply(trans2,axis=1)
```

In [39]:
```
df_needtrans['translate']
df_needtrans.to_csv('translate.csv',index=False,sep=',',encoding='utf_8_sig')
```

In [45]:
```
df_origin_text=df_movies[df_movies['retweet_id']=='None']
```

In [ ]:
```
df_tran_text=df_origin_text.merge(df_needtrans, how='left',on='Unnamed: 0')
df_tran_text
```

In [80]:
```
# Do sentiment Analysis
from textblob import TextBlob


df_tran_text['translate']=df_tran_text['translate'].fillna('English')
df_tran_text
df_tran_text['general']=df_tran_text[['deal_x','translate']].apply(lambda x:
                                                                   x[0] if x[1]==
'English' \
                                                                   else x[1],axis
=1)
```

In [81]:
```
def sentiment(text):
    blob = TextBlob(text)
    return(blob.sentiment[0])
df_tran_text['sentiment']=df_tran_text['general'].apply(sentiment)
```

In [83]:
```
df_tran_text.to_csv("sentiment.csv",index=False,sep=',',encoding='utf_8_sig')
```

In [1]:

```
import pandas as pd
import numpy as np
```

In [3]:

```
df=pd.read_csv("movie_data2.csv")
```

```
/Users/ruining/anaconda3/lib/python3.7/site-packages/IPython/core/interact
iveshell.py:3057: DtypeWarning: Columns (10) have mixed types. Specify dty
pe option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [4]:

```
films=["Wonderstruck","TheSnowman","SameKindOfDifferentAsMe","Geostorm","Boo2",\
       "OnlyTheBrave","TheKillingOfASacredDeer","Leatherface","Novitiate",\
       "Suburbicon","TheWork","JungleMovie","BadDayForTheCut","TragedyGirls",\
       "KillingGunther","Neverhere","KillsOnWheels","TheBachelors","SamuraiJack",\
       "TheForeigner","HappyDeathDay","AmericanSatan","BreatheMovie","OverdriveMovie",\
       "BloodMoneyMovie","GnomeAlone","GoodbyeChristopherRobin","MarshallMovie",\
       "ProfessorMarston"]
```

In [5]:

```
films_lower=[]
for film in films:
    films_lower.append(film.lower())
```

In [6]:

```
import re
n=len(df['hashtags'])
hashtags_new=[""]*n
for index in range(n):
    if (df['hashtags'][index]=='None'):
        hashtags_list=[]
        text=df['text'][index].split()
        for words in text:
            if words[0]=="#":
                hashtags_list.append(words[1:-1])
        string=" "
        for words in hashtags_list:
            if words.lower() in films_lower:
                string=words+string
        if len(string)>1:
            hashtags_new[index]=string[:-1]
        else:
            hashtags_new[index]=" "
    elif df['hashtags'][index]=="[]":
        hashtags_new[index]=" "
    else:
        text=re.sub(r'[^\w]',' ',df['hashtags'][index])
        string=" "
        for words in text.split():
            if words.lower() in films_lower:
                string=words+string
        if len(string)>1:
            hashtags_new[index]=string[:-1]
        else:
            hashtags_new[index]=" "
```

In [69]:

```
new_df=df.copy()
new_df['hashtags']=hashtags_new
```

In [70]:

```
new_df.to_csv("movie_data3.csv",index=False)
```

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
#with open('.\movie_data3\movie_data3.csv', 'r',encoding='utf-8', newline='')
 as csvfile:
#with open('.\movie_data3\movie_data3.csv', 'r', encoding='utf-8', newline='')
as csvfile:
df=pd.read_csv('.\movie_data3\movie_data3.csv',encoding = "ISO-8859-1",low_mem
ory=False)
```

In [ ]:
```
films=["Wonderstruck","TheSnowman","SameKindOfDifferentAsMe","Geostorm","Boo2"
,"OnlyTheBrave","TheKillingOfASacredDeer",
       "Leatherface","Novitiate","Suburbicon","TheWork","JungleMovie","BadDayF
orTheCut","TragedyGirls","KillingGunther",
       "Neverhere","KillsOnWheels","TheBachelors","SamuraiJack","TheForeigner"
,"HappyDeathDay","AmericanSatan","BreatheMovie",
       "OverdriveMovie","BloodMoneyMovie","GnomeAlone","GoodbyeChristopherRobi
n","MarshallMovie","ProfessorMarston"]
# films=[Wonderstruck   TheSnowman   samekindofdifferentasme   geostorm   boo2
OnlytheBrave   TheKillingofaSacredDeer
#        LEATHERFACE   Novitiate   Suburbicon   TheWork   junglemovie   BadDay
ForTheCut   TragedyGirls   KillingGunther
#        neverhere   KillsOnWheels   TheBachelors   samuraijack   theforeigner
happydeathday   americansatan   BreatheMovie
#        overdrivemovie   bloodmoneymovie   GnomeAlone   GoodbyeChristopherRob
in   Marshallmovie   ProfessorMarston]
```

In [ ]:
```
films_lower=[]
for film in films:
    films_lower.append(film.lower())
identi=df['Unnamed: 0'].values.tolist()
```

In [ ]:
```
df_tag=pd.DataFrame(0,index=identi,columns=films_lower)
```

In [ ]:
```
df=df.set_index('Unnamed: 0')
n=len(df['hashtags'])
for i in identi:
    tags=str(df['hashtags'][i]).lower().split(' ')
    for tag in tags:
        if tag in films_lower:
            df_tag[tag][i]=1
```

In [ ]:
```
df_tag=df_tag.fillna(0)
def s(x):
    s=0
    for i in x:
        s+=int(i)
    return s
df_tag['sum']=df_tag.apply(s,axis=1)
df_tag['sum']
```

In [ ]:
```
df_tag=df_tag.reset_index()
df_tag.rename(columns={'index': 'Unnamed: 0'})
df[['Unnamed: 0','hashtags']]
df.merge(df_tag,how='left',on='Unnamed: 0')
df_tag=df_tag.rename(columns={'index': 'Unnamed: 0'})
df_summerge=df.merge(df_tag,how='left',on='Unnamed: 0')
df_summerge.columns

#df_tag['sum']
```

In [ ]:
```
df_tag.to_csv("tags.csv",index=True,sep=','
```

In [7]:

```
import pandas as pd
import numpy as np
```

In [10]:

```
df=pd.read_csv('movie_data2 (1).csv')
location=list(df.location.unique())
location.sort()
len(location)
```

Out[10]:

29628

In [17]:

```
uslist=['usa','USA','US','us','United States','United States of America',\
        'united states','united states of america','america','America','American']
fllist=['FL','FLORIDA','florida','Florida','Fort Myers','Miami']
allist=['alabama','Alabama','Huntsville']
aklist=['AK','ALASKA','Alaska','alaska']
azlist=['AZ','ARIZONA','arizona','Arizona']
arlist=['AR','Arkansas']
calist=['California','CA','california','Cali','cali','Santa Monica','Los Angeles','Sili
con Valley','Hollywood','Bay Area',\
       'Berkeley','Beverly Hills','L.A','LA','San Francisco','West Covina','BayArea','C
SU','LOS ANGELES']
colist=['CO','COLORADO','Colorado','Denver']
ctlist=['CT','CONNECTICUT','Connecticut']
delist=['DE','DELAWARE','delaware','Delaware']
galist=['GA','GEORGIA','georgia','Georgia','ATL','Atlanta']
hilist=['HI','HAWAII','hawaii','Hawaii']
idlist=['ID','IDAHO','idaho','Idaho']
illist=['IL','ILLINOIS','illinois','Illinois','Chicago','chicago']
inlist=['IN','INDIANA','indiana','Indiana']
ialist=['IA','IOWA','iowa','Iowa']
kslist=['KS','KANSAS','kansas','Kansas']
kylist=['KY','KENTUCKY','kentucky','Kentucky']
lalist=['LOUISIANA','Louisiana','louisiana','Baton Rouge','New Orleans']
melist=['ME','MAINE','Maine','maine']
mdlist=['MD','MARYLAND','Maryland','maryland']
malist=['Massachusetts','MASSACHUSETTS','massachusetts','Boston','Belmont']
milist=['Michigan','michigan','MICHIGAN','MI','Detroit']
mnlist=['MN','MINNESOTA','minnesota','Minnesota']
mslist=['MS','Mississippi','MISSISSIPPI','mississippi']
molist=['MO','MISSOURI','Missouri','missouri','Kansas City']
mtlist=['MT','MONTANA','Montana','montana']
nelist=['NE','Nebraska','nebreska']
nvlist=['NV','NEVADA','nevada','Nevada','Las Vegas']
nhlist=['NH','New Hampshire','NEW HAMPSHIRE','New hampshire','new hampshire','New Hamps
hre']
njlist=['NJ','New Jersey','NEW JERSEY','new jersey','New jersey']
nmlist=['NM','New Mexico','new mexico','NEW MEXICO','new mexico']
nylist=['NY','NEW YORK','New york','new york','New York','Brooklyn','Bronx','NYC','New
 Yawk']
nclist=['NC','North Carolina','NORTH CAROLINA','north carolina']
ndlist=['ND','North Dakota','north dakota','North dakota']
ohlist=['OH','Ohio','OHIO','ohio','Cincinnati']
oklist=['OK','Oklahoma','OKLAHOMA','oklahoma']
orlist=['OREGON','oregon','Oregon']
palist=['PA','PENNSYLVANIA','Pennsylvania','pennsylvania','PennsylvanNIA','Hyrule','Phi
ladelphia','Pittsburgh']
rilist=['RI','Rhode Island','rhode island','Rhode island']
sdlist=['SD','SOUTH DAKOTA','south dakota','South Dakota']
sclist=['SC','South Carolina','SOUTH CAROLINA','south carolina']
tnlist=['TN','TENNESSEE','Tennessee','tennessee','Chattanooga','Nashville']
txlist=['TX','TEXAS','Texas','texas','Dallas','Austin','htx']
utlist=['UT','UTAH','utah','Utah']
vtlist=['VT','Vermont','VERMONT','vermont']
valist=['Virginia','VIRGINIA','virginia','Arlington, VA','Williamsburg, VA']
walist=['Washington, USA','Seattle','Washington State']
wvlist=['WV','WEST VIRGINIA','west virginia','West virginia']
wilist=['WI','WISCONSIN','Wisconsin','wisconsin','Milwaukee']
wylist=['WY','Wyoming','WYOMING','wyoming']
dc=['DC','D.C']
```

```
pr=['Puerto Rico']


india = ['India','Bangalore','Bombay','chennai','Chennai','Chodavaram','Bogor','Mumbai'
,'Vizag']
uk = ['UK','Britain','England','London','United Kingdom','UnitedKingdom','Scotland','Ir
eland','Belfast',\
     'Bournemouth','bournemouth','Buckinghamshire','cornwall','Cornwall','Devon','Dubli
n','Manchester',\
     'Birmingham','Cardiff','england','Liverpool','Wales']
japan= ['Japan','日本','オ','お','ん','大阪','神','犬小屋','東京']
canada = ['Canada','CANADA','Toronto','British Columbia','Calgary','Québec','Montreal',
'Vancouver',\
          'TORONTO','Ontario','Lethbridge','Manitoba']
sa=['South Africa']
brasil=['Brasil','Belo Horizonte','Brazil']
qatar=['Qatar']
venezuela =['Venezuela']
thai =['Bangkok','Thailand','TH','thailand','BKK','Chiang Mai','Chula','Nakhon Ratchasi
ma','กรุงเทพมหานคร',\
      'จ.เชียงใหม่','ประเทศไทย','น','Phuket','bangkok','bkk']
phil=['Republic of the Philippines','Philippines','Cagayan Valley','Calabarzon','Genera
l Santos City','Manila','Quezon City']
singapore=['Singapore']
spain=['Spain','Alicante','Barcelona','España','Zaragoza','Valladolid','Valencia','Cata
lunya','Madrid']
deutsch=['Deutschland','Berlin','Germany']
turkey=['Turkey']
portugal=['Portugal']
peru=['Peru','Lima']
italy=['Italy','Bassano del Grappa','Lazio','Roma','italia','Italia']
argentina =['Argentina','Buenos Aire']
greece=['Greece','Cyprus','Ελλάς']
newz=['New Zealand']
aust=['Australia','Brisbane','Queensland','Melbourne']
indo= ['Indonesia','Bandung','bandung','Klaten','Medan','Yogyakarta','Jakarta']
belgium=['Belgium','Belgique']
colombia=['Bogotá','Bogota']
france=['France','Paris','Bourges','france']
hungary=['Hungary','hungary','Budapest']

denmark=['Denmark']
mexico=['CDMX','ciudad de mexico','Ciudad de México','México','Mexico']
china=['China','香港','台灣','北京','中华人民共和国']
uae=['Dubai','UAE','United Arab Emirates']
middle=['Saudi Arabia','المملكة الأردنية الهاشمية','Cairo','Egypt','يمكن تعجبك المفضلة','Lebanon']
korea=['Korea','태국']
malay=['Kuala','Kuantan','Kuching','Malaysia','MALAYSIA','malaysia']
kuwait=['Kuwait']
nigeria=['Nigeria']
norway=['Norway']
finland=['Finland']
nether =['Netherlands']
```

In [18]:

```
filteredArrayusa = [i for i in location if any(i for j in uslist if str(j) in i)]
filteredArrayfl = [i for i in location if any(i for j in fllist if str(j) in i)]
filteredArrayal = [i for i in location if any(i for j in allist if str(j) in i)]
filteredArrayak = [i for i in location if any(i for j in aklist if str(j) in i)]
filteredArrayaz = [i for i in location if any(i for j in azlist if str(j) in i)]
filteredArrayar = [i for i in location if any(i for j in arlist if str(j) in i)]
filteredArrayca = [i for i in location if any(i for j in calist if str(j) in i)]
filteredArrayco = [i for i in location if any(i for j in colist if str(j) in i)]
filteredArrayct = [i for i in location if any(i for j in ctlist if str(j) in i)]
filteredArrayde = [i for i in location if any(i for j in delist if str(j) in i)]
filteredArrayga = [i for i in location if any(i for j in galist if str(j) in i)]
filteredArrayhi = [i for i in location if any(i for j in hilist if str(j) in i)]
filteredArrayid = [i for i in location if any(i for j in idlist if str(j) in i)]
filteredArrayil = [i for i in location if any(i for j in illist if str(j) in i)]
filteredArrayin = [i for i in location if any(i for j in inlist if str(j) in i)]
filteredArrayia = [i for i in location if any(i for j in ialist if str(j) in i)]
filteredArrayks = [i for i in location if any(i for j in kslist if str(j) in i)]
filteredArrayky = [i for i in location if any(i for j in kylist if str(j) in i)]
filteredArrayla = [i for i in location if any(i for j in lalist if str(j) in i)]
filteredArrayme = [i for i in location if any(i for j in melist if str(j) in i)]
filteredArraymd = [i for i in location if any(i for j in mdlist if str(j) in i)]
filteredArrayma = [i for i in location if any(i for j in malist if str(j) in i)]
filteredArraymi = [i for i in location if any(i for j in milist if str(j) in i)]
filteredArraymn = [i for i in location if any(i for j in mnlist if str(j) in i)]
filteredArrayms = [i for i in location if any(i for j in mslist if str(j) in i)]
filteredArraymo = [i for i in location if any(i for j in molist if str(j) in i)]
filteredArraymt = [i for i in location if any(i for j in mtlist if str(j) in i)]
filteredArrayne = [i for i in location if any(i for j in nelist if str(j) in i)]
filteredArraynv = [i for i in location if any(i for j in nvlist if str(j) in i)]
filteredArraynh = [i for i in location if any(i for j in nhlist if str(j) in i)]
filteredArraynj = [i for i in location if any(i for j in njlist if str(j) in i)]
filteredArraynm = [i for i in location if any(i for j in nmlist if str(j) in i)]
filteredArrayny = [i for i in location if any(i for j in nylist if str(j) in i)]
filteredArraync = [i for i in location if any(i for j in nclist if str(j) in i)]
filteredArraynd = [i for i in location if any(i for j in ndlist if str(j) in i)]
filteredArrayoh = [i for i in location if any(i for j in ohlist if str(j) in i)]
filteredArrayok = [i for i in location if any(i for j in oklist if str(j) in i)]
filteredArrayor = [i for i in location if any(i for j in orlist if str(j) in i)]
filteredArraypa = [i for i in location if any(i for j in palist if str(j) in i)]
filteredArrayri = [i for i in location if any(i for j in rilist if str(j) in i)]
filteredArraysd = [i for i in location if any(i for j in sdlist if str(j) in i)]
filteredArraysc = [i for i in location if any(i for j in sclist if str(j) in i)]
filteredArraytn = [i for i in location if any(i for j in tnlist if str(j) in i)]
filteredArraytx = [i for i in location if any(i for j in txlist if str(j) in i)]
filteredArrayut = [i for i in location if any(i for j in utlist if str(j) in i)]
filteredArrayvt = [i for i in location if any(i for j in vtlist if str(j) in i)]
filteredArrayva = [i for i in location if any(i for j in valist if str(j) in i)]
filteredArraywa = [i for i in location if any(i for j in walist if str(j) in i)]
filteredArraywv = [i for i in location if any(i for j in wvlist if str(j) in i)]
filteredArraywi = [i for i in location if any(i for j in wilist if str(j) in i)]
filteredArraywy = [i for i in location if any(i for j in wylist if str(j) in i)]
filteredArraydc = [i for i in location if any(i for j in dc if str(j) in i)]
filteredArraypr = [i for i in location if any(i for j in pr if str(j) in i)]



filteredArrayindia = [i for i in location if any(i for j in india if str(j) in i)]
filteredArrayuk = [i for i in location if any(i for j in uk if str(j) in i)]
filteredArrayjapan = [i for i in location if any(i for j in japan if str(j) in i)]
```

```
filteredArraycanada = [i for i in location if any(i for j in canada if str(j) in i)]
filteredArraysa = [i for i in location if any(i for j in sa if str(j) in i)]
filteredArraybrasil = [i for i in location if any(i for j in brasil if str(j) in i)]
filteredArrayqatar = [i for i in location if any(i for j in qatar if str(j) in i)]
filteredArrayvene = [i for i in location if any(i for j in venezuela if str(j) in i)]
filteredArraythai = [i for i in location if any(i for j in thai if str(j) in i)]
filteredArrayphil = [i for i in location if any(i for j in phil if str(j) in i)]
filteredArraysing = [i for i in location if any(i for j in singapore if str(j) in i)]
filteredArrayspain = [i for i in location if any(i for j in spain if str(j) in i)]
filteredArraydeutsch = [i for i in location if any(i for j in deutsch if str(j) in i)]
filteredArrayturkey = [i for i in location if any(i for j in turkey if str(j) in i)]
filteredArrayportugal = [i for i in location if any(i for j in portugal if str(j) in i
)]
filteredArrayperu = [i for i in location if any(i for j in peru if str(j) in i)]
filteredArrayitaly = [i for i in location if any(i for j in italy if str(j) in i)]
filteredArrayargen = [i for i in location if any(i for j in argentina if str(j) in i)]
filteredArraygreece = [i for i in location if any(i for j in greece if str(j) in i)]
filteredArraynewz = [i for i in location if any(i for j in newz if str(j) in i)]
filteredArrayaust = [i for i in location if any(i for j in aust if str(j) in i)]
filteredArrayindo = [i for i in location if any(i for j in indo if str(j) in i)]
filteredArraybelgium = [i for i in location if any(i for j in belgium if str(j) in i)]
filteredArraycolombia = [i for i in location if any(i for j in colombia if str(j) in i
)]
filteredArrayfrance = [i for i in location if any(i for j in france if str(j) in i)]
filteredArrayhungary = [i for i in location if any(i for j in hungary if str(j) in i)]
filteredArraydenmark = [i for i in location if any(i for j in denmark if str(j) in i)]
filteredArraymexico = [i for i in location if any(i for j in mexico if str(j) in i)]
filteredArraychina = [i for i in location if any(i for j in china if str(j) in i)]
filteredArrayuae = [i for i in location if any(i for j in uae if str(j) in i)]
filteredArraymiddle = [i for i in location if any(i for j in middle if str(j) in i)]
filteredArraykorea = [i for i in location if any(i for j in korea if str(j) in i)]
filteredArraymalay = [i for i in location if any(i for j in malay if str(j) in i)]
filteredArraykuwait = [i for i in location if any(i for j in kuwait if str(j) in i)]
filteredArraynigeria = [i for i in location if any(i for j in nigeria if str(j) in i)]
filteredArraynorway = [i for i in location if any(i for j in norway if str(j) in i)]
filteredArrayfinland =[i for i in location if any(i for j in finland if str(j) in i)]
filteredArraynether =[i for i in location if any(i for j in nether if str(j) in i)]
```

In [19]:

```
filteredArrayall = filteredArrayfl+filteredArrayal+filteredArrayak+filteredArrayaz+\
filteredArrayar+filteredArrayca+filteredArrayco+filteredArrayct+filteredArrayde+\
filteredArrayga+filteredArrayhi+filteredArrayid+filteredArrayil+filteredArrayin+\
filteredArrayia+filteredArrayks+filteredArrayky+filteredArrayla+filteredArrayme+\
filteredArraymd+filteredArrayma+filteredArraymi+filteredArraymn+filteredArrayms+\
filteredArraymo+filteredArraymt+filteredArrayne+filteredArraynv+filteredArraynh+\
filteredArraynj+filteredArraynm+filteredArrayny+filteredArraync+filteredArraynd+\
filteredArrayoh+filteredArrayok+filteredArrayor+filteredArraypa+filteredArrayri+\
filteredArraysd+filteredArraysc+filteredArraytn+filteredArraytx+filteredArrayut+\
filteredArrayvt+filteredArrayva+filteredArraywa+filteredArraywv+filteredArraywi+\
filteredArraywy+filteredArraydc+filteredArraypr+filteredArrayindia+filteredArrayuk+filt
eredArrayjapan+\
filteredArraycanada+filteredArraysa+filteredArraybrasil+filteredArrayqatar+filteredArra
yvene+\
filteredArraythai+filteredArrayphil+filteredArraysing+filteredArrayspain+filteredArrayd
eutsch+\
filteredArrayturkey+filteredArrayportugal+filteredArrayperu+filteredArrayitaly+filtered
Arrayargen+\
filteredArraygreece+filteredArraynewz+filteredArrayaust+filteredArrayindo+filteredArray
belgium+\
filteredArraycolombia+filteredArrayfrance+filteredArrayhungary+filteredArraydenmark+fil
teredArraymexico+\
filteredArraychina+filteredArrayuae+filteredArraymiddle+filteredArraykorea+filteredArra
ymalay+\
filteredArraykuwait+filteredArraynigeria+filteredArraynorway+filteredArrayfinland+filte
redArraynether
```

In [20]:

```
listall=[filteredArrayfl]+[filteredArrayal]+[filteredArrayak]+[filteredArrayaz]+\
[filteredArrayar]+[filteredArrayca]+[filteredArrayco]+[filteredArrayct]+[filteredArrayde]+\
[filteredArrayga]+[filteredArrayhi]+[filteredArrayid]+[filteredArrayil]+[filteredArrayin]+\
[filteredArrayia]+[filteredArrayks]+[filteredArrayky]+[filteredArrayla]+[filteredArrayme]+\
[filteredArraymd]+[filteredArrayma]+[filteredArraymi]+[filteredArraymn]+[filteredArrayms]+\
[filteredArraymo]+[filteredArraymt]+[filteredArrayne]+[filteredArraynv]+[filteredArraynh]+\
[filteredArraynj]+[filteredArraynm]+[filteredArrayny]+[filteredArraync]+[filteredArraynd]+\
[filteredArrayoh]+[filteredArrayok]+[filteredArrayor]+[filteredArraypa]+[filteredArrayri]+\
[filteredArraysd]+[filteredArraysc]+[filteredArraytn]+[filteredArraytx]+[filteredArrayut]+\
[filteredArrayvt]+[filteredArrayva]+[filteredArraywa]+[filteredArraywv]+[filteredArraywi]+\
[filteredArraywy]+[filteredArraydc]+[filteredArraypr]+\
[filteredArrayindia]+[filteredArrayuk]+[filteredArrayjapan]+\
[filteredArraycanada]+[filteredArraysa]+[filteredArraybrasil]+[filteredArrayqatar]+[filteredArrayvene]+\
[filteredArraythai]+[filteredArrayphil]+[filteredArraysing]+[filteredArrayspain]+[filteredArraydeutsch]+\
[filteredArrayturkey]+[filteredArrayportugal]+[filteredArrayperu]+[filteredArrayitaly]+[filteredArrayargen]+\
[filteredArraygreece]+[filteredArraynewz]+[filteredArrayaust]+[filteredArrayindo]+[filteredArraybelgium]+\
[filteredArraycolombia]+[filteredArrayfrance]+[filteredArrayhungary]+[filteredArraydenmark]+[filteredArraymexico]+\
[filteredArraychina]+[filteredArrayuae]+[filteredArraymiddle]+[filteredArraykorea]+[filteredArraymalay]+\
[filteredArraykuwait]+[filteredArraynigeria]+[filteredArraynorway]+[filteredArrayfinland]+[filteredArraynether]
```

In [21]:

```
names=['FL','AL','AK','AZ','AR','CA','CO','CT','DE','GA','HI','ID','IL','IN','IA','KS','KY','LA','ME',\
       'MD','MA','MI','MN','MS','MO','MT','NE','NV','NH','NJ','NM','NY','NC','ND','OH','OK','OR','PA',\
       'RI','SD','SC','TN','TX','UT','VT','VA','WA','WV','WI','WY','Washington, DC','Puerto Rico','India','UK',\
       'Japan','Canada','South Africa','Brasil','Qatar','Venezuela','Thailand','Philippines','Singapore','Spain',\
       'Germany','Turkey','Portugal','Peru',\
       'Italy','Argentina','Greece','New Zealand','Australia','Indonesia','Belgium','Columbia','France',\
      'Hungary','Denmark','Mexico','China','UAE','Middle East','Korea','Malaysia','Kuwait','Nigeria','Norway',\
      'Finland','Netherlands']
```

In [22]:

```
df['new_location']=''
for i in range(df.shape[0]):
    if df.iloc[i,11] in filteredArrayall:
        for k in range(90):
            if df.iloc[i,11] in listall[k]:
                df.iloc[i,-1]=names[k]
```

In [23]:

```
filteredArraycountry = filteredArrayusa+filteredArrayfl+filteredArrayal+filteredArrayak
+filteredArrayaz+\
filteredArrayar+filteredArrayca+filteredArrayco+filteredArrayct+filteredArrayde+\
filteredArrayga+filteredArrayhi+filteredArrayid+filteredArrayil+filteredArrayin+\
filteredArrayia+filteredArrayks+filteredArrayky+filteredArrayla+filteredArrayme+\
filteredArraymd+filteredArrayma+filteredArraymi+filteredArraymn+filteredArrayms+\
filteredArraymo+filteredArraymt+filteredArrayne+filteredArraynv+filteredArraynh+\
filteredArraynj+filteredArraynm+filteredArrayny+filteredArraync+filteredArraynd+\
filteredArrayoh+filteredArrayok+filteredArrayor+filteredArraypa+filteredArrayri+\
filteredArraysd+filteredArraysc+filteredArraytn+filteredArraytx+filteredArrayut+\
filteredArrayvt+filteredArrayva+filteredArraywa+filteredArraywv+filteredArraywi+\
filteredArraywy+filteredArraydc+filteredArraypr+filteredArrayindia+filteredArrayuk+filt
eredArrayjapan+\
filteredArraycanada+filteredArraysa+filteredArraybrasil+filteredArrayqatar+filteredArra
yvene+\
filteredArraythai+filteredArrayphil+filteredArraysing+filteredArrayspain+filteredArrayd
eutsch+\
filteredArrayturkey+filteredArrayportugal+filteredArrayperu+filteredArrayitaly+filtered
Arrayargen+\
filteredArraygreece+filteredArraynewz+filteredArrayaust+filteredArrayindo+filteredArray
belgium+\
filteredArraycolombia+filteredArrayfrance+filteredArrayhungary+filteredArraydenmark+fil
teredArraymexico+\
filteredArraychina+filteredArrayuae+filteredArraymiddle+filteredArraykorea+filteredArra
ymalay+\
filteredArraykuwait+filteredArraynigeria+filteredArraynorway+filteredArrayfinland+filte
redArraynether

listcountry=[filteredArrayusa]+[filteredArrayfl]+[filteredArrayal]+[filteredArrayak]+[f
ilteredArrayaz]+\
[filteredArrayar]+[filteredArrayca]+[filteredArrayco]+[filteredArrayct]+[filteredArrayd
e]+\
[filteredArrayga]+[filteredArrayhi]+[filteredArrayid]+[filteredArrayil]+[filteredArrayi
n]+\
[filteredArrayia]+[filteredArrayks]+[filteredArrayky]+[filteredArrayla]+[filteredArraym
e]+\
[filteredArraymd]+[filteredArrayma]+[filteredArraymi]+[filteredArraymn]+[filteredArraym
s]+\
[filteredArraymo]+[filteredArraymt]+[filteredArrayne]+[filteredArraynv]+[filteredArrayn
h]+\
[filteredArraynj]+[filteredArraynm]+[filteredArrayny]+[filteredArraync]+[filteredArrayn
d]+\
[filteredArrayoh]+[filteredArrayok]+[filteredArrayor]+[filteredArraypa]+[filteredArrayr
i]+\
[filteredArraysd]+[filteredArraysc]+[filteredArraytn]+[filteredArraytx]+[filteredArrayu
t]+\
[filteredArrayvt]+[filteredArrayva]+[filteredArraywa]+[filteredArraywv]+[filteredArrayw
i]+\
[filteredArraywy]+[filteredArraydc]+[filteredArraypr]+\
[filteredArrayindia]+[filteredArrayuk]+[filteredArrayjapan]+\
[filteredArraycanada]+[filteredArraysa]+[filteredArraybrasil]+[filteredArrayqatar]+[fil
teredArrayvene]+\
[filteredArraythai]+[filteredArrayphil]+[filteredArraysing]+[filteredArrayspain]+[filte
redArraydeutsch]+\
[filteredArrayturkey]+[filteredArrayportugal]+[filteredArrayperu]+[filteredArrayitaly]+
[filteredArrayargen]+\
[filteredArraygreece]+[filteredArraynewz]+[filteredArrayaust]+[filteredArrayindo]+[filt
eredArraybelgium]+\
[filteredArraycolombia]+[filteredArrayfrance]+[filteredArrayhungary]+[filteredArraydenm
```

```
ark]+[filteredArraymexico]+\
[filteredArraychina]+[filteredArrayuae]+[filteredArraymiddle]+[filteredArraykorea]+[fil
teredArraymalay]+\
[filteredArraykuwait]+[filteredArraynigeria]+[filteredArraynorway]+[filteredArrayfinlan
d]+[filteredArraynether]
```

In [24]:

```
country=['USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','USA',
'USA','USA',\
       'USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','U
SA',\
       'USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','U
SA',\
       'USA','USA','USA','USA','USA','USA','USA','USA','USA','USA','India','UK',\
       'Japan','Canada','South Africa','Brasil','Qatar','Venezuela','Thailand','Philipp
ines','Singapore','Spain',\
       'Germany','Turkey','Portugal','Peru',\
       'Italy','Argentina','Greece','New Zealand','Australia','Indonesia','Belgium','Co
lumbia','France',\
      'Hungary','Denmark','Mexico','China','UAE','Middle East','Korea','Malaysia','Kuwa
it','Nigeria','Norway',\
      'Finland','Netherlands']
```

In [25]:

```
df['new_country']=''
for i in range(df.shape[0]):
    if df.iloc[i,11] in filteredArraycountry:
        for k in range(91):
            if df.iloc[i,11] in listcountry[k]:
                df.iloc[i,-1]=country[k]
```

In [201]:

```
df.to_csv('new_country1.csv',sep=',', encoding='utf-8')
```

In [ ]:

```

```

Sentiment

12/8/2019

```
tweets=read.csv("Follower.csv")
```

```
names(tweets)
```

```
##  [1] "Unnamed..0"      "id_x"            "text"
##  [4] "retweet_count"   "reply_count"     "like_count"
##  [7] "dayofweek"       "Month"           "date"
## [10] "time"            "head"            "Time"
## [13] "Year"            "hashtags"        "username"
## [16] "userid"          "followers_count" "location_x"
## [19] "description"     "retweet_id"      "deal_x"
## [22] "id_y"            "deal_y"          "lan"
## [25] "translate"       "general"         "sentiment"
## [28] "hashtag"         "location_y"      "new_location"
## [31] "new_country"
```

```
tweets_info=tweets[,c("retweet_count","dayofweek","Month","date","Time","new_location","followers_count
```

```
tweets_info=tweets_info[tweets_info$hashtag!="",]
tweets_info=tweets_info[tweets_info$new_location!="",]
tweets_info=tweets_info[tweets_info$new_country=="USA",]
```

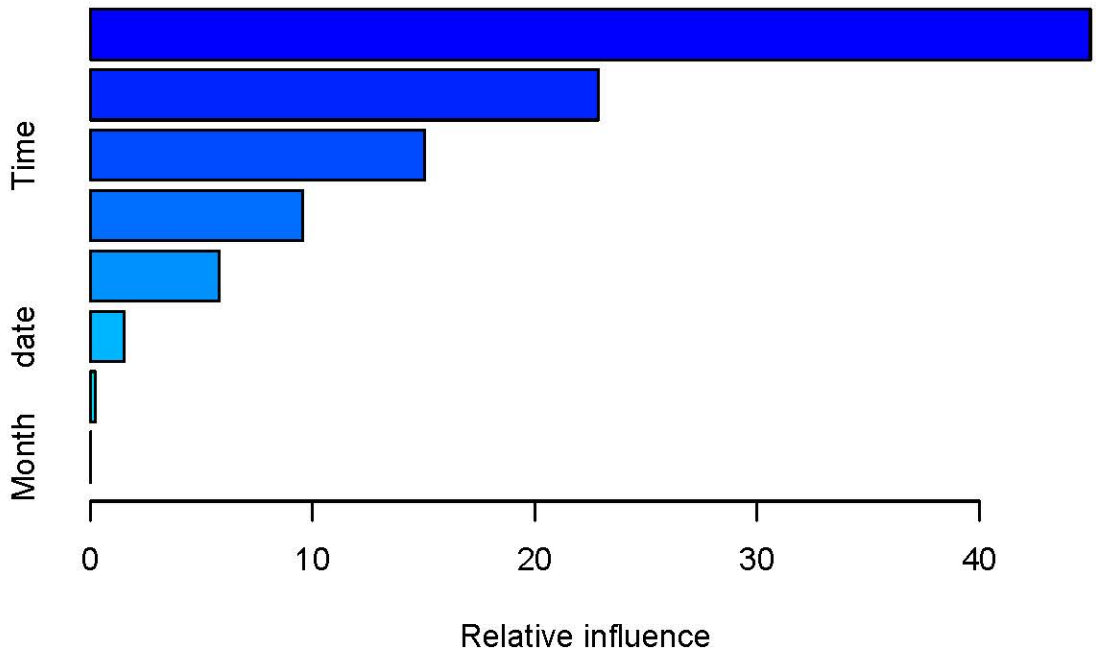
```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
set.seed(1)

boost.gno = gbm(sentiment~.-new_country,tweets_info,distribution = "gaussian", n.trees=5000,interaction

summary(boost.gno)
```




```
##                            var     rel.inf
## new_location       new_location 44.99501318
## hashtag                 hashtag 22.86599030
## Time                       Time 15.02209248
## followers_count followers_count  9.55645796
## dayofweek             dayofweek  5.79137692
## date                       date  1.51946576
## retweet_count     retweet_count  0.23376511
## Month                     Month  0.01583829
```