Hotel Tone Analyzer:

Auther: Esraa Madi

Overview

Social media and online review sites encourage a ton of writing about your business or brand. With so much valuable data hidden in plain sight, you can't let it just sit there! we need to mine large samples of unstructured data easily with emotion detection and sentiment analysis.

Interpreting a text is not an easy thing to do. showing you how much anger, joy, fear, disgust, or sadness are likely in the text needs a lot of processes to detect emotions.

A quick example, the sentence, "Sorry, can't make it." has a 50% likelihood of sadness. But if you change the period to an exclamation point, the likelihood of sadness drops to 38% and anger becomes the most dominant emotion.

Tone Analyzer is a service determines the emotional tone behind a series of words, used to gain an understanding of the attitudes, opinions and emotions expressed within an online mention.

This Service Makes Sure You'll Never Misinterpret A Text Again!

It is critical for any organization to understand and track the general sentiment of its users at any given time.

Here we going to use one of common emotional or sentiment analysis APIs **IBM Tone Analyzer** to analysis text reviews of 1,000 hotels and get the total emotional tones for a hotel

IBM Tone Analyzer (https://www.ibm.com/watson/services/tone-analyzer/)

is one of the cognitive services provided by IBM Cloud. It can help us predict the emotions, tones and communication style of the text written by users (passed as input to the service).

To have an insight on the service and tones, try this web interface (https://tone-analyzer-demo.ng.bluemix.net/).

How Hotel Tone Analyzer Service Works?

It calculate the normalized total tones for the hotel using (Watson python lib). This lib will give you a normalized score for the detected tones, then we will aggregate them all and get a final score.

For example, if we have for a specific hotel:

- Review #1 scored 0.25 angry, and 0.80 sad
- Review #2 scored 0.7 happy, and 0.65 sad
- Review #3 scored 0.2 happy, 0.7 angry, and 0.4 sad

So the total normalized tones for this hotel is 0.47 angry, 0.45 happy, and 0.62 sad

---

Datasets

The dataset used in this project is taken from this Kaggle dataset (https://www.kaggle.com/datafiniti/hotel-reviews#7282_1.csv)

This dataset is a list of about 1,000 hotels and 30,000 reviews. The dataset includes hotel location, name, rating, review data, title, username, and more.

For more information about dataset columns, you can check this data dictionary (https://developer.datafiniti.co/docs/business-data-schema)

Let's start :)

In [9]:

```
# import libraries
import pandas as pd
import numpy as np
```

In [119]:

```
# read hotel data
places_df = pd.read_csv("../data/raw_data/7282_1.csv")
```

In [120]:

```
# check first 5 rows of dataframe
places_df.head()
```

Out[120]:

	address	categories	city	country	latitude	longitude	name	postalCode	provinc
0	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
1	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
2	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
3	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
4	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA

In [121]:

```
# check data shape
places_df.shape
```

Out[121]:

```
(35912, 19)
```

We have here almost 36,000 reviews for hotels

In [122]:

```
# clean columns names ( '.' will raise an error during calling dataframe column
s)
places_df.columns = [col.replace('.', '_') for col in places_df.columns]
```

In [123]:

```
# checking missing values
places_df.isnull().sum()
```

Out[123]:

```
address                     0
categories                  0
city                        0
country                     0
latitude                   86
longitude                  86
name                        0
postalCode                 55
province                    0
reviews_date              259
reviews_dateAdded           0
reviews_doRecommend     35912
reviews_id              35912
reviews_rating            862
reviews_text               22
reviews_title            1622
reviews_userCity        19649
reviews_username           43
reviews_userProvince    18394
dtype: int64
```

As we can see here, there is a lot of missing values in the hotel info and review details. I don't think I am going to use in my analyzer more than **reviews text** and **hotel name**

For the **hotel name** column, there are no missing values.

For the **review text** there are only 22 missing reviews over a total of 35912 , we will delete them.

In [124]:

```
# check sample of reviews
places_df[['name', 'reviews_text']].head()
```

Out[124]:

	name	reviews_text
0	Hotel Russo Palace	Pleasant 10 min walk along the sea front to th...
1	Hotel Russo Palace	Really lovely hotel. Stayed on the very top fl...
2	Hotel Russo Palace	Ett mycket bra hotell. Det som drog ner betyge...
3	Hotel Russo Palace	We stayed here for four nights in October. The...
4	Hotel Russo Palace	We stayed here for four nights in October. The...

In [125]:

```
print('Our data contains {} for {} (hotels, resorts or restaurants) '.format(pla
ces_df.shape[0],len(places_df.name.unique())))
```

```
Our data contains 35912 for 879 (hotels, resorts or restaurants)
```

In [126]:

```
# filter data with "Hotels" value in "categories" column
places_df.categories.value_counts()[:10]
```

Out[126]:

```
Hotels
21420
Hotels,Hotel
2977
Hotel,Hotels
1524
Hotels,Hotels & Motels
423
Hotels,Casinos
392
Banquet Rooms,Hotels,Banquet Facilities,Hotels & Motels,Hotel,Hotels
Motels                                                    320
Travel & Transport,Hotels,Hotels & Motels
319
Conventions Conferences & Trade Shows,Wedding Receptions & Parties,C
onvention & Meeting Facilities & Services,Hotels      314
Hotels & Motels
305
Wedding Sites,Hotels,Hotels & Motels,Meeting Facilities
271
Name: categories, dtype: int64
```

As we can see the reviewed places in the dataset are classified as `hotels` or `hotels and casinos` or `hotels and restaurants` at the same time, so we will filter any place with `hotels` word in his category

In [127]:

```
def check_hotel_category(cat_list):
    '''
    Function takes full categories list for any place and return 1 if is hotel
    *Args:
        cat_list : string, place categories
    *Return:
        integer, 1 if the place is hotel, otherwise 0
    '''
    for cat in cat_list.split(','):
        if cat.lower() == 'hotels':
            return 1
    return 0
```

In [128]:

```
# add new column called is_hotel to use it later on filtering
places_df['is_hotel'] = places_df.categories.map(check_hotel_category)
```

In [129]:

```
# create a new dataframe with hotels only
hotels_df = places_df[places_df.is_hotel == 1].copy()
```

In [130]:

```
# check how many hotels we have now
hotels_df.shape
```

Out[130]:

```
(33625, 20)
```

In [131]:

```
# check hotels dataframe header
hotels_df.head()
```

Out[131]:

	address	categories	city	country	latitude	longitude	name	postalCode	provinc
0	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
1	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
2	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
3	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
4	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA

In [132]:

```
# check missing values again
hotels_df.isnull().sum()
```

Out[132]:

```
address                     0
categories                  0
city                        0
country                     0
latitude                   82
longitude                  82
name                        0
postalCode                 55
province                    0
reviews_date              233
reviews_dateAdded           0
reviews_doRecommend     33625
reviews_id              33625
reviews_rating            491
reviews_text               21
reviews_title            1171
reviews_userCity        18932
reviews_username           41
reviews_userProvince    17615
is_hotel                    0
dtype: int64
```

In [133]:

```
# Delete those 21 rows, since the API works on text mainly
hotels_df.drop(hotels_df[hotels_df.reviews_text.isnull()].index, inplace=True)
```

IBM Watson python lib

The library has two functions we can use it to do tone analysis:

1. Analyze general tone:
   - This service analyze the content for emotional and language tones. It can analyzes the tone of the full document; by default,
2. Analyze customer-engagement tone
   - This service analyze the tone of customer service and support conversations. It can help you better understand your interactions with customers and improve your communications in general or for specific customers.

I will use the first one since it matches my problem here more

There are two versions of Api:

- 2016-05-19 :
- 2017-09-21 :
  - The service can return results for the following tone IDs:
    - Emotional tones: anger, fear, joy, and sadness.
    - Language tones: analytical, confident, and tentative.
  - The service returns results only for tones whose scores meet a minimum threshold of 0.5.

I will use the most recent version.

Installation

To get started with IBM Tone Analyzer and get required credentials, I found this link (https://github.com/watson-developer-cloud/python-sdk) is very helpful.

In [134]:

```
# install first
#!pip install --upgrade ibm-watson
```

Supplying the API key and version

In [135]:

```
# import Api libraries
import json
from ibm_watson import ToneAnalyzerV3
from ibm_watson.tone_analyzer_v3 import ToneInput
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
```

In [136]:

```
# set key, version and service url
authenticator = IAMAuthenticator("")
service = ToneAnalyzerV3(version='2017-09-21', authenticator=authenticator)
service.set_service_url("https://gateway-lon.watsonplatform.net/tone-analyzer/api")
```

Text cleaning

As far as I checked, I could not find anything about text cleaning before using IBM tone analyzer API, but I found in the API documentation that they consider in their tone prediction features such as punctuation, emoticons, language parameters such as sentence structure, and sentence complexity.
So I don't think I am going to remove them since they have some kind of impact on the predicted tone.
Another reason to keep them is the general tone API uses punctuation to do sentence segmentation.

Get tones for each review text

The response of general tone service API contains two predicted tone analysis:

- document-level analysis
- sentence-level analysis.

I will use document-level analysis which represents (review-level) in my case then merge these predicted document tones beside each review in our dataframe.
Regarding the full response, we will store it in a list then save in a file later for further use.

In [137]:

```
# save api response in list
review_tone_analysis = []
```

In [138]:

```
# concat document-level analysis to dataframe by adding a column for each possib
le returned tone and set default value 0 hotels_df['review_anger'] = [0]*hotels_
df.shape[0]
hotels_df['review_fear'] = [0]*hotels_df.shape[0]
hotels_df['review_joy'] = [0]*hotels_df.shape[0]
hotels_df['review_sadness'] = [0]*hotels_df.shape[0]
hotels_df['review_analytical'] = [0]*hotels_df.shape[0]
hotels_df['review_confident'] = [0]*hotels_df.shape[0]
hotels_df['review_tentative'] = [0]*hotels_df.shape[0]
```

In [139]:

```
# check datafram header
hotels_df.head(2)
```

Out[139]:

	address	categories	city	country	latitude	longitude	name	postalCode	provinc
0	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
1	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA

2 rows × 26 columns

In [346]:

```
print('Our data contains {} for {} (hotels, resorts or restaurants) '.format(hot
els_df.shape[0],len(hotels_df.name.unique())))
```

```
Our data contains 33604 for 725 (hotels, resorts or restaurants)
```

> We will loop over 33604 review and get the tone from each one.

In [348]:

```
def get_text_tone(txt):
    '''
    Function takes text and use IBM tone analyser to get the text tone
    *Args:
        txt: string, any text need to get its tone
    *Return:
        tone_analysis: dict, containt document-level and sentence-level analysis
for the text
    '''

    tone_analysis = service.tone({'text': txt}, content_type="text/plain").get_r
esult()

    return tone_analysis
```

In [358]:

```
# loop over hotels reviews
for index, row in hotels_df.iterrows():

    # get review tone analysis
    review_tone = get_text_tone(row.reviews_text)

    # save response in the list
    review_tone_analysis.append(review_tone)

    # loop over returned (document-level) tones of the text
    for tone in review_tone['document_tone']['tones']:

        # edit obtained tone in the dataframe
        col_name = 'review_' + tone['tone_id']
        hotels_df.loc[index, col_name] = tone['score']
```

In [361]:

```
# since the API requests took almost 10 hours, we save the current version of da
taframe in the file
hotels_df.to_csv('../data/processed_data/hotel_review_text_tone.csv')
```

In [362]:

```
# save the full API response in the file just in case we need it later
api_response = pd.DataFrame({'tone':review_tone_analysis})
api_response.to_csv('../data/processed_data/review_text_tone_response.csv')
```

Get tones for each review title

Usually, in the review title users tell things which left the most impacts on them either they like it or not. So I think doing analysis for the review title itself maybe will add some value.

In [365]:

```
# check missing values on review title
hotels_df[hotels_df.reviews_title.isnull()].shape
```

Out[365]:

```
(1153, 27)
```

In [ ]:

```
# fill missing titles with ' '
hotels_df.reviews_title.fillna(' ', inplace=True)
```

In [255]:

```
# save api response in list
title_tone_analysis = []
```

In [145]:

```
# concat document-level analysis to dataframe by adding a column for each possib
le returned tone and set defult value 0
hotels_df['review_title_anger'] = [0]*hotels_df.shape[0]
hotels_df['review_title_fear'] = [0]*hotels_df.shape[0]
hotels_df['review_title_joy'] = [0]*hotels_df.shape[0]
hotels_df['review_title_sadness'] = [0]*hotels_df.shape[0]
hotels_df['review_title_analytical'] = [0]*hotels_df.shape[0]
hotels_df['review_title_confident'] = [0]*hotels_df.shape[0]
hotels_df['review_title_tentative'] = [0]*hotels_df.shape[0]
```

In [146]:

```
# check datafram header
hotels_df.head(2)
```

Out[146]:

	address	categories	city	country	latitude	longitude	name	postalCode	provinc
0	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
1	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA

2 rows × 34 columns

> We will loop over review titles and get the tone from each one

In [358]:

```
# loop over hotels reviews titles
for index, row in hotels_df.iterrows():

    # check if number of title words > 3, since api limit for input text is mini
mum 3 words
    if len(row.reviews_title.split(' ')) > 3 :

        # get review title tone analysis
        review_tone = get_text_tone(row.reviews_title)

        # save response in the list
        title_tone_analysis.append(review_tone)

        # loop over returned (document-level) tones of the text
        for tone in review_tone['document_tone']['tones']:

            # edit obtained tone in the dataframe
            col_name = 'review_title_' + tone['tone_id']
            hotels_df.loc[index, col_name] = tone['score']
```

In [361]:

```
# save the current version of dataframe in the file
hotels_df.to_csv('../data/processed_data/hotel_review_text_tone.csv')
```

In [362]:

```
# save the full API response in the file just in case we need it later
api_response = pd.DataFrame({'tone':review_tone_analysis})
api_response.to_csv('../data/processed_data/review_title_tone_response.csv')
```

Get Normalized Total Tones For Each Hotel

In [59]:

```
# reload processed data
hotels_df = pd.read_csv('../data/processed_data/hotel_review_text_tone.csv', ind
ex_col=0)
```

In [60]:

```
# check data header
hotels_df.head(2)
```

Out[60]:

	address	categories	city	country	latitude	longitude	name	postalCode	provinc
0	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA
1	Riviera San Nicol 11/a	Hotels	Mableton	US	45.421611	12.376187	Hotel Russo Palace	30126	GA

2 rows × 34 columns

In [ ]:

```
# aggregate reviews tones for each hotel
```

In [63]:

```
def tone_score_avg(col):
    '''
    Function to caluclate normalized score for each hotel (the average of tone s
core)
    *Args:
        col: series, given column
    *Return:
        avg: float, average of tones without non-zero numbers
    '''

    sum_ = sum(col.tolist()) # total sum
    count_ = np.count_nonzero(col.tolist()) # count only non zero cells
    try:
        avg = sum_ / count_
    except:
        avg = 0
    return avg
```

In [64]:

```
# get normalized score of reviews for each hotel
hotels_norm_tone = hotels_df.groupby('name', as_index=False).agg({
                                    'review_anger': [tone_score_avg]
                                  , 'review_fear': [tone_score_avg]
                                  , 'review_joy': [tone_score_avg]
                                  , 'review_sadness': [tone_score_avg]
                                  , 'review_analytical': [tone_score_avg]
                                  , 'review_confident': [tone_score_avg]
                                  , 'review_tentative': [tone_score_avg]
                                  , 'review_title_anger': [tone_score_avg]
                                  , 'review_title_fear': [tone_score_avg]
                                  , 'review_title_joy': [tone_score_avg]
                                  , 'review_title_sadness': [tone_score_avg]
                                  , 'review_title_analytical': [tone_score_avg]
                                  , 'review_title_confident': [tone_score_avg]
                                  , 'review_title_tentative': [tone_score_avg]

})

# fix columns names
hotels_norm_tone.columns = hotels_norm_tone.columns.droplevel(1)
```

In [66]:

```
# check normalized score for hotels
hotels_norm_tone.head(2)
```

Out[66]:

	name	review_anger	review_fear	review_joy	review_sadness	review_analytical	review
0	40 Berkeley Hostel	0.704641	0.638483	0.690031	0.607036	0.697771	
1	A Bed & Breakfast In Cambridge	0.586374	0.608599	0.743332	0.518749	0.721827	

In [67]:

```
# save result into file to be able to use it later in flask app
hotels_norm_tone.to_csv('../data/processed_data/hotels_norm_tone.csv')
```

Test normalized score for one hotel

We will get two normalized score tones:

- Review text tones
- Review title tones

Let's test `40 Berkeley Hostel` Hotel

In [10]:

```
# import plotting libraries
from plotly import graph_objs as go
import matplotlib.pyplot as plt
import plotly.offline as py
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
py.init_notebook_mode(connected=True)
```

In [11]:

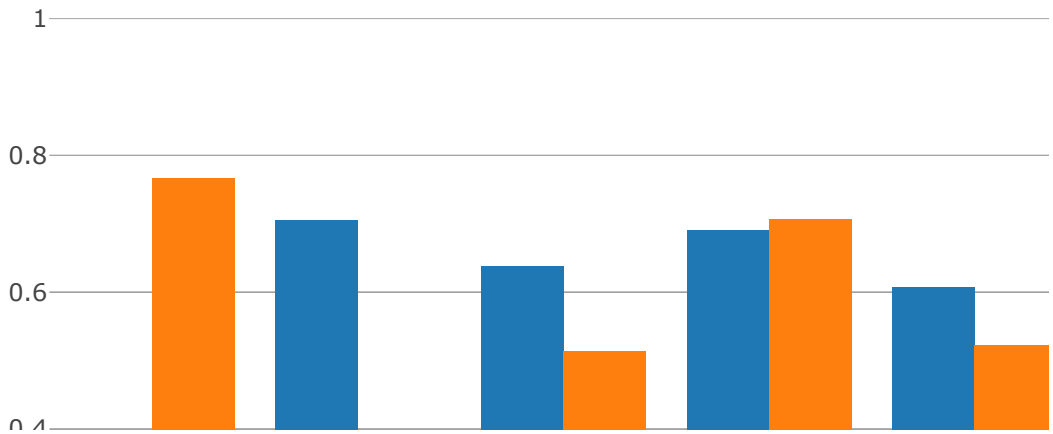
```
hotels_norm_tone = pd.read_csv('../data/processed_data/hotels_norm_tone.csv')
```

In [15]:

```
def get_hotel_tone(hotel_name):
    '''
    Function to get and plot normalized score for given hotel (review text tones
and review title tones)
    *Args:
        hotel_name: string, given hotel name
    *Return:
    '''

    # get hotel normalized score
    hotel_tone = pd.Series(hotels_norm_tone[hotels_norm_tone.name == hotel_name]
.iloc[:,1:].to_dict(orient='records')[0])

    # plot tones
    fig = go.Figure(data=[
    go.Bar(name='Review Text Tones', x=[i.replace('review_','') for i in hotel_t
one.index[:7]], y=hotel_tone.values[:7]),
    go.Bar(name='Review Title Tones', x=[i.replace('review_','') for i in hotel_
tone.index[:7]], y=hotel_tone.values[7:])])

    # Change the bar mode
    fig.layout.update(barmode='group')
    py.iplot(fig)
    #fig.show()
```

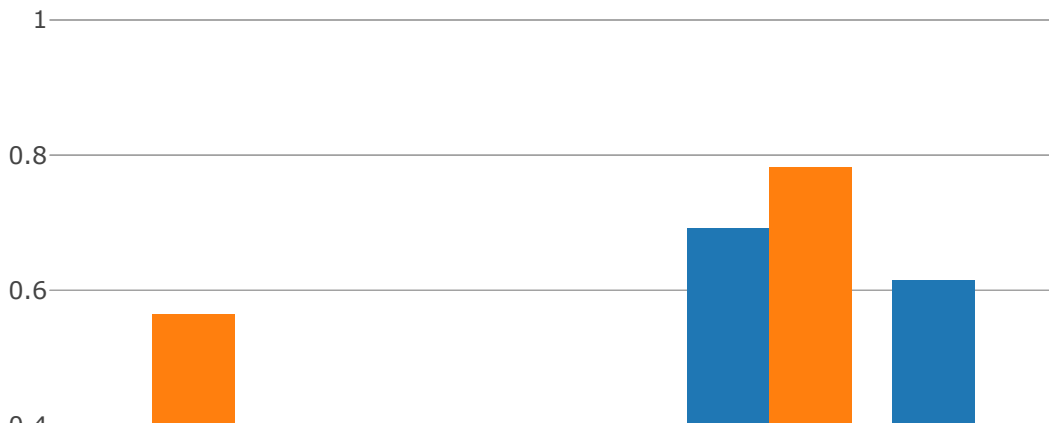
In [16]:

```
# test '40 Berkeley Hostel'
get_hotel_tone('40 Berkeley Hostel')
```



In [17]:

```
# test 'La Quinta'
get_hotel_tone('La Quinta')
```



As we mentioned above, we have calculated review title tones to see how much it reflects the tones in the text of reviews.

We can see in cases like `40 Berkeley Hostel` with almost 160 reviews, there is no significant difference between reviews text tones and reviews title tones.

For `La Quinta` hotel with 7 reviews, the reviews title tones are different than reviews text tones.