Oops... Nothing Here.. So, you are on your own this time.

Import Libraries

In [1]:
```python
!pip install wget
!pip install twython
import wget
import sys
print(sys.version)

#Plot
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from scipy.stats.kde import gaussian_kde

#Data Packages
import math
import pandas
import numpy as np

#Progress bar
from tqdm import tqdm

#Counter
from collections import Counter

#Operation
import operator

#Natural Language Processing Packages
import re
import nltk
from nltk.corpus import stopwords

## Download Resources
nltk.download("vader_lexicon")
nltk.download("stopwords")
nltk.download("averaged_perceptron_tagger")
nltk.download("wordnet")

from nltk.sentiment import SentimentAnalyzer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.sentiment.util import *
from nltk import tokenize
from nltk.corpus import stopwords
from nltk.tag import PerceptronTagger
from nltk.data import find

## Machine Learning
import sklearn
import sklearn.metrics as metrics
```

```
Requirement already satisfied: wget in /usr/local/lib/python3.6/dist-packages (3.2)
Requirement already satisfied: twython in /usr/local/lib/python3.6/dist-packages (3.7.0)
Requirement already satisfied: requests>=2.1.0 in /usr/local/lib/python3.6/dist-packages (from t
wython) (2.21.0)
Requirement already satisfied: requests-oauthlib>=0.4.0 in /usr/local/lib/python3.6/dist-package
s (from twython) (1.3.0)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages
(from requests>=2.1.0->twython) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packages (from re
quests>=2.1.0->twython) (2.8)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (fro
m requests>=2.1.0->twython) (2019.9.11)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages
(from requests>=2.1.0->twython) (1.24.3)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.6/dist-packages (from r
equests-oauthlib>=0.4.0->twython) (3.1.0)
3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0]
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [0]:
```
# wget.download("https://github.com/MIE451-1513-2019/course-datasets/raw/master/reviews.zip")
# !unzip reviews.zip
```

Load Data

In [0]:
```
#Read in from pandas
hotelDf = pandas.read_csv('reviews.csv')
hotelDf.columns=['filePath','hotelName','reviewColumn','ratingScore','groundTruth']
```

In [4]:
```
hotelDf.head()
```

Out[4]:

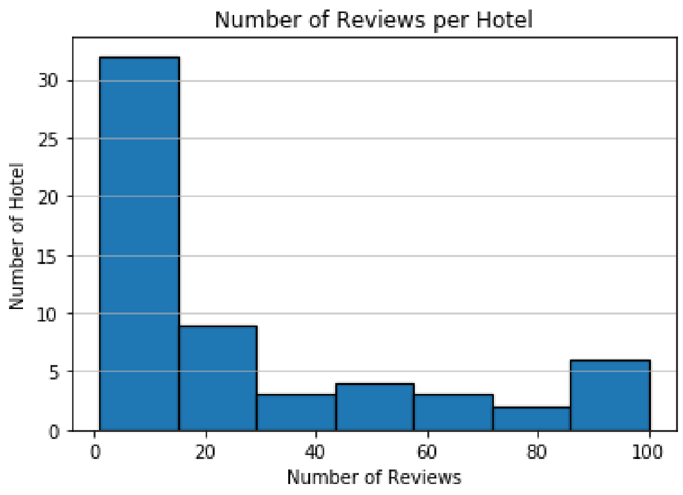
	filePath	hotelName	reviewColumn	ratingScore	groundTruth
0	data/ca/297653/10089626/452253954.html	Hotel Divine Valley	"We were 3 families that drove to Panchgani. W...	4	positive
1	data/ca/297653/10089626/531008588.html	Hotel Divine Valley	"we were 5 ppl and a kid who stayed there..we ...	2	negative
2	data/ca/297653/10089626/581399063.html	Hotel Divine Valley	"Villas are spacious and comfortable.. Clubhou...	5	positive
3	data/ca/297653/10243112/375397733.html	The Arowana Villa	"Peaceful, relaxing, beautiful, cozy... I can'...	4	positive
4	data/ca/297653/10243112/478268268.html	The Arowana Villa	"We 3 families stayed at Arowana Panchgani be...	4	positive

Q1. Sentiment Analysis and Aggregation

Q1(a)

```
In [0]: def getNoOfRatings(hotelDf):
          ratings=hotelDf.groupby('hotelName').size().to_frame(name='no_of_reviews').reset_index()
          plt.hist(ratings.no_of_reviews.values, bins='auto', edgecolor='black', linewidth=1.2)
          plt.grid(axis='y', alpha=0.75)
          plt.xlabel('Number of Reviews')
          plt.ylabel('Number of Hotel')
          plt.title('Number of Reviews per Hotel')
          return ratings
```

```
In [6]: ratings=getNoOfRatings(hotelDf)
```



```
In [7]: ratings.describe()
```

Out[7]:

	no_of_reviews
count	59.000000
mean	27.694915
std	31.985586
min	1.000000
25%	5.000000
50%	12.000000
75%	44.500000
max	100.000000

In [0]:
```
# Use vader to evaluated sentiment of reviews
def evalSentences(sentences, to_df=False, columns=[]):
    #Instantiate an instance to access SentimentIntensityAnalyzer class
    sid = SentimentIntensityAnalyzer()
    pdlist = []
    if to_df:
        for sentence in tqdm(sentences):
            ss = sid.polarity_scores(sentence)
            pdlist.append([sentence]+[ss['compound']])
        reviewDf = pandas.DataFrame(pdlist)
        reviewDf.columns = columns
        return reviewDf

    else:
        for sentence in tqdm(sentences):
            print(sentence)
            ss = sid.polarity_scores(sentence)
            for k in sorted(ss):
                print('{0}: {1}, '.format(k, ss[k]), end='')
            print()
```

In [9]:
```
reviews = hotelDf['reviewColumn'].values
reviewDF = evalSentences(reviews, to_df=True, columns=['reviewCol','vader'])
reviewDF.head()
```

```
100%|██████████| 1634/1634 [00:02<00:00, 767.31it/s]
```

Out[9]:

	reviewCol	vader
0	"We were 3 families that drove to Panchgani. W...	0.9961
1	"we were 5 ppl and a kid who stayed there..we ...	0.3887
2	"Villas are spacious and comfortable.. Clubhou...	0.8221
3	"Peaceful, relaxing, beautiful, cozy... I can'...	0.9952
4	"We 3 families stayed at Arowana Panchgani be...	0.9924

In [0]:
```python
# Note: You may want to use an NLTK tokenizer instead of a regular expression in the following
def dataFrameTransformation(hotelDf, reviewDF, k=500):
    reviews = reviewDF['reviewCol'].values

    stop = set(stopwords.words('english'))

    # Top-k frequent terms
    counter = Counter()
    for review in reviews:
            counter.update([word.lower()
                            for word
                            in re.findall(r'\w+', review)
                            if word.lower() not in stop and len(word) > 2])
    topk = counter.most_common(k)

    #Find out if a particular review has the word from topk list
    freqReview = []
    for i in range(len(reviews)):
        tempCounter = Counter([word.lower() for word in re.findall(r'\w+',reviews[i])])
        topkinReview = [1 if tempCounter[word] > 0 else 0 for (word,wordCount) in topk]
        freqReview.append(topkinReview)


    #Prepare freqReviewDf
    freqReviewDf = pandas.DataFrame(freqReview)
    dfName = []
    for c in topk:
        dfName.append(c[0])
    freqReviewDf.columns = dfName
    finalreviewDf = reviewDF.join(freqReviewDf)
    finaldf = hotelDf[['hotelName','ratingScore','groundTruth']].join(finalreviewDf)
    return topk, finaldf
```

In [11]:
```
topk, finaldf = dataFrameTransformation(hotelDf, reviewDF, k=500)
finaldf.head()
```

Out[11]:

	hotelName	ratingScore	groundTruth	reviewCol	vader	hotel	good	food	room	rooms	place	stay	staf
0	Hotel Divine Valley	4	positive	"We were 3 families that drove to Panchgani. W...	0.9961	0	1	1	1	0	1	1	1
1	Hotel Divine Valley	2	negative	"we were 5 ppl and a kid who stayed there..we ...	0.3887	0	1	0	1	0	1	0	0
2	Hotel Divine Valley	5	positive	"Villas are spacious and comfortable.. Clubhou...	0.8221	0	0	0	0	0	0	1	0
3	The Arowana Villa	4	positive	"Peaceful, relaxing, beautiful, cozy... I can'...	0.9952	0	1	0	0	1	0	0	0
4	The Arowana Villa	4	positive	"We 3 families stayed at Arowana Panchgani be...	0.9924	0	1	0	0	0	1	0	1

5 rows × 505 columns

In [12]:
```
#Extract a list of hotels
hotelNames = finaldf['hotelName'].unique()
hotelNames
```

Out[12]:
```
array(['Hotel Divine Valley', 'The Arowana Villa', 'Ashirwad Bungalow',
       'Summer Plaza Resort', 'Mount View Heritage Hotel', 'Mount Castle',
       'OYO 10483 Hotel The Signature Crest', 'Blue Country Resort',
       'Hotel Millennium Park', 'Casa Bella Resort',
       'SaffronStays Verandah by the Valley', 'Bellevue Resort',
       'Hotel Summer Palace', 'Prospect Hotel',
       'Quick Getaway Clubs &amp; Resorts', 'The Loft', 'Miraya Hotel',
       'OYO 9566 Hotel JK Excellency', 'Hotel Blossom, Panchgani',
       'Animish Bungalow', 'Maitri Retreat', 'Umbrella Inn',
       'Hotel Gitanjali', 'Il Palazzo Hotel', 'Tranquil Treasure',
       'Royal Orbit', 'Lodge Suvidha', 'Magnus Caverns Resort',
       'Treebo Trend Prince Palace', 'Basilica Rediscover Serenity',
       'Mount View Executive, The Valley Resort', 'Terra Camp Resort',
       'Jivanta Hotel', 'Rahil Plaza', 'Brickland Hotel',
       'Sharda Arogya Dham', 'Hotel Mala&#39;s', 'The Dhanhills',
       'Exotic Home Stay - Panchgani', 'Hira Baug', 'Gurukripa Bungalow',
       'Rainforest Restaurant &amp; Villas',
       'JenJon Holiday Homes Panchgani', 'Hotel River Palace Panchgani',
       'Mayur Agro Park', 'Bhatia Villas', 'Ravine Hotel', 'S2 Residency',
       'Trinity House', 'OYO 22372 Sherbaug- A Theme Park And Resort',
       'Elysium Resort Panchgani', 'Hotel Residency', 'Silver Oak Villa',
       'Raval Bunglow', 'Panchgani Tent House', 'Hotel Silver Leef',
       'Alliance Tents and Accommodations', 'Hotel Pan Hill',
       'Hotel Valley Nest'], dtype=object)
```

In [0]:
```
def getHotelRank(df, measure='ratingScore'):
    #Rank the hotel by ground truth rating score
    hotelRating = []
    for hotel in hotelNames:
        itemDf = df.loc[df['hotelName']==hotel]
        hotelRating.append([hotel,itemDf[measure].mean()])
    hotelRatingDfGt = pandas.DataFrame(hotelRating)
    hotelRatingDfGt.columns=['hotelName','avgRatingScore']
    hotelRatingDfGt = hotelRatingDfGt.sort_values('avgRatingScore',ascending=0)
    return hotelRatingDfGt
```

In [14]:
```
hotelRatingDfGt = getHotelRank(finaldf)
hotelRatingDfGt.head()
```

Out[14]:

	hotelName	avgRatingScore
29	Basilica Rediscover Serenity	5.000000
15	The Loft	5.000000
24	Tranquil Treasure	5.000000
32	Jivanta Hotel	5.000000
27	Magnus Caverns Resort	4.833333

```
In [15]: hotelRatingDfVd = getHotelRank(finaldf, measure='vader')
         hotelRatingDfVd.head()
```

Out[15]:

	hotelName	avgRatingScore
32	Jivanta Hotel	0.991800
24	Tranquil Treasure	0.979100
42	JenJon Holiday Homes Panchgani	0.977350
44	Mayur Agro Park	0.973267
10	SaffronStays Verandah by the Valley	0.964688

Q1 (b)

```
In [16]: hotelRatingDfGt = getHotelRank(finaldf)
         hotelRatingDfGt.head()
```

Out[16]:

	hotelName	avgRatingScore
29	Basilica Rediscover Serenity	5.000000
15	The Loft	5.000000
24	Tranquil Treasure	5.000000
32	Jivanta Hotel	5.000000
27	Magnus Caverns Resort	4.833333

```
In [17]: hotelRatingDfVd = getHotelRank(finaldf, measure='vader')
         hotelRatingDfVd.head()
```

Out[17]:

	hotelName	avgRatingScore
32	Jivanta Hotel	0.991800
24	Tranquil Treasure	0.979100
42	JenJon Holiday Homes Panchgani	0.977350
44	Mayur Agro Park	0.973267
10	SaffronStays Verandah by the Valley	0.964688

```
In [18]: for review in finaldf.loc[finaldf['hotelName']=='The Loft']['reviewCol']:
           print(review)
```

```
"Breathtakingly beautiful place to get away from the hustle bustle of  the concrete jungle....\n
Super host loving and generous very good rooms clean and hygenic...Very impressive \nAwesome foo
d thoroughly enjoyed..\n\n
"Went with family and had a wonderful time. Overlooks the valley and lake. Rooms are well appoin
ted and luxurious. Breakfast to die for ! The hosts go all out to make sure you have a great tim
e. You won't be disappointed with this place.
"Clean property, excellent rooms. The food was exceptionally great,Pitambar the chef prepared so
me of the most exceptional food I've ever eaten. It's funny how you meet such exceptional talent
in such remote cities.
"Visited during the monsoons which made it perfect for the breathtaking view this place has to o
ffer.\nThe rooms were neatly kept and well desingned with a mini kitchen to facilitate all of ou
r needs. We didn't bother cooking though cause the in house chef made the most delectable meals 
even entertaining our demands(tantrums!)\nThe place was being managed by a certain Mr.Shivam- en
ergetic young chap and super hospitable who helped arrange local transport and other commodities
as per our request.\nTo sum it up - this place made me feel home away from home.
"The loft has well maintained rooms which are spacious combined with a kitchen. Quiet and very b
reezy that you would  not need AC even in the summer heat. The view is also good. The caretakers
take good care and provide homely food as per your order. I would most definitely recommend this
place and come back again.  Unfortunately there were some generator issues when we were to check
in but Shivam did a phenomenal job of eliminating any discomfort on our part and went the extra 
mile.
"We had a great stay here. The service was excellent. It\u2019s located 4-5mins from Panchgani m
arket and has superb view. The food served here too was excellent. It\u2019s a no-brainer for me
the next time I plan to visit Panchgani.
"Pitambar is the man. Him & kamlabai made sure we were were never short of anything!\n\nThe room
s are really well done and the view from the property is very pretty. That it is a bit away from
the main market really calms the place down.\n\nThe food here just stands out. Trust their recom
mendations, and eat local!
```

```
In [0]: analyser = SentimentIntensityAnalyzer()
        def sentiment_analyzer_scores(sentence):
            score = analyser.polarity_scores(sentence)
            print("{:-<40} {}".format(sentence, str(score)))
```

```
In [20]: sentiment_analyzer_scores("Breakfast to die for !")
```

```
Breakfast to die for !------------------ {'neg': 0.583, 'neu': 0.417, 'pos': 0.0, 'compound': -
0.636}
```

Answer 1(b):
Two of the Five Hotels agree on the mention in Top-5. To investigate, one of the top-5 hotel- 'The Loft' as per the average ground truth rating as taken. The review for this hotel had tokens such as the 'die'. The vader lexicon rating for these tokens is very low (-2.9) [https://github.com/cjhutto/vaderSentiment/blob/master/vaderSentiment/vader_lexicon.txt (https://github.com/cjhutto/vaderSentiment/blob/master/vaderSentiment/vader_lexicon.txt)]. This lead to an overall lower score for this hotel.

Q2. Frequency Analysis

Q2 (a)

```
In [0]: #We are only intereseted in this three column for overall analysis
        itemAnalysisDf = finaldf[['reviewCol','groundTruth','vader']]
```

```
In [0]: def getTopK(df, k, label_value, label_column='groundTruth', operation=operator.eq, value_column
        ='reviewCol'):
            stop = set(stopwords.words('english'))
            #Add possible Stop Words for Hotel Reviews
            stop.add('hotel')
            stop.add('room')
            stop.add('rooms')
            stop.add('stay')
            stop.add('staff')
            counter = Counter()
            for review in df.loc[operation(df[label_column],label_value)][value_column]:
                    counter.update([word.lower()
                                    for word
                                    in re.findall(r'\w+', review)
                                    if word.lower() not in stop and len(word) > 2])
            topk = counter.most_common(k)
            return topk
```

```
In [23]: topkGroundPos = getTopK(df=itemAnalysisDf, k=50, label_value='positive')
         topkGroundPos
```

```
Out[23]: [('good', 1127),
          ('food', 1124),
          ('place', 936),
          ('panchgani', 587),
          ('view', 586),
          ('well', 529),
          ('clean', 470),
          ('family', 435),
          ('also', 430),
          ('one', 425),
          ('nice', 421),
          ('nthe', 420),
          ('service', 414),
          ('best', 373),
          ('property', 372),
          ('time', 371),
          ('great', 365),
          ('pool', 322),
          ('visit', 316),
          ('resort', 312),
          ('valley', 300),
          ('would', 296),
          ('like', 283),
          ('area', 266),
          ('stayed', 264),
          ('experience', 257),
          ('location', 253),
          ('beautiful', 240),
          ('maintained', 240),
          ('excellent', 239),
          ('really', 236),
          ('away', 220),
          ('breakfast', 216),
          ('market', 211),
          ('get', 208),
          ('restaurant', 202),
          ('helpful', 202),
          ('amazing', 191),
          ('swimming', 184),
          ('home', 181),
          ('table', 179),
          ('spacious', 178),
          ('around', 175),
          ('old', 170),
          ('quite', 167),
          ('big', 167),
          ('parsi', 166),
          ('small', 163),
          ('comfortable', 162),
          ('day', 160)]
```

```
In [24]: topkGroundNeg = getTopK(df=itemAnalysisDf, k=50, label_value='negative')
         topkGroundNeg
```

```
Out[24]: [('good', 400),
          ('food', 307),
          ('place', 250),
          ('one', 193),
          ('water', 183),
          ('service', 175),
          ('view', 150),
          ('nthe', 149),
          ('also', 146),
          ('pool', 141),
          ('panchgani', 138),
          ('like', 127),
          ('even', 126),
          ('location', 123),
          ('would', 122),
          ('property', 121),
          ('clean', 120),
          ('well', 115),
          ('time', 115),
          ('resort', 113),
          ('breakfast', 108),
          ('small', 107),
          ('bad', 105),
          ('night', 103),
          ('get', 102),
          ('bathroom', 100),
          ('area', 96),
          ('stayed', 93),
          ('experience', 93),
          ('booked', 93),
          ('day', 92),
          ('available', 91),
          ('better', 88),
          ('could', 88),
          ('people', 84),
          ('booking', 80),
          ('restaurant', 80),
          ('around', 80),
          ('money', 80),
          ('reception', 80),
          ('much', 78),
          ('maintained', 77),
          ('great', 77),
          ('dirty', 77),
          ('bed', 77),
          ('table', 75),
          ('old', 74),
          ('hot', 74),
          ('nice', 74),
          ('really', 73)]
```

```
In [25]: finaldf.groundTruth.value_counts()
```

```
Out[25]: positive    1233
         negative     401
         Name: groundTruth, dtype: int64
```

Answer 2(a):

- Topk or top50 positive and negative words were chosen based on the frequency in a positive and negative reviews respectively
- The topK positive tokens such as the 'view', 'valley' suggests that it is a scenic place
- The token such as 'food' is decieving because they appear frequently in positive as well as negative reviews. However, off the total of 1233 positive reviews (91%), 'food' appears 1124 times and of the total of 401 negative reviews, 'food' appears 307 times (76%). Thus, it can be claimed that overall people like food at Pachgini
- People liked 'parsi' point at this place
- People are not happy with the 'water' at this place

Q2 (b)

In [0]:
```
# Noun Phrase Extraction Support Functions
stopwords_g = stopwords.words('english')
lemmatizer = nltk.WordNetLemmatizer()
stemmer = nltk.stem.porter.PorterStemmer()

# generator, generate leaves one by one
def leaves(tree):
    """Finds NP (nounphrase) leaf nodes of a chunk tree."""
    for subtree in tree.subtrees(filter = lambda t: t.label()=='NP' or t.label()=='JJ' or t.lab
el()=='RB'):
        yield subtree.leaves()

# stemming, lematizing, lower case...
def normalise(word):
    """Normalises words to lowercase and stems and lemmatizes it."""
    word = word.lower()
    word = stemmer.stem(word)
    word = lemmatizer.lemmatize(word)
    return word

# stop-words and length control
def acceptable_word(word):
    """Checks conditions for acceptable word: length, stopword."""
    accepted = bool(2 <= len(word) <= 40
        and word.lower() not in stopwords_g)
    return accepted

# generator, create item once a time
def get_terms(tree):
    for leaf in leaves(tree):
        term = [normalise(w) for w,t in leaf if acceptable_word(w) ]
        # Phrase only
        if len(term)>1:
            yield term
```

In [0]:
```
# Flatten phrase lists to get tokens for analysis
def flatten(npTokenList):
    finalList =[]
    for phrase in npTokenList:
        token = ''
        for word in phrase:
            token += word + ' '
        finalList.append(token.rstrip())
    return finalList
```

In [0]:
```
# Revise the previous dataframe transform function...
def newDataFrameTransformation(hotelDf, reviewDF, k=50):
    reviews = reviewDF['reviewCol'].values

    # Top-k frequent terms
    counter = Counter()
    for review in reviews:
            counter.update(flatten([word
                            for word
                            in get_terms(chunker.parse(pos_tag(re.findall(r'\w+', review))))
                            ]))
    topk = counter.most_common(k)

    #Find out if a particular review has the word from topk list
    freqReview = []
    for i in range(len(reviews)):
        tempCounter = Counter(flatten([word
                                       for word
                                       in get_terms(chunker.parse(pos_tag(re.findall(r'\w+',rev
iews[i]))))]))
        topkinReview = [1 if tempCounter[word] > 0 else 0 for (word,wordCount) in topk]
        freqReview.append(topkinReview)


    #Prepare freqReviewDf
    freqReviewDf = pandas.DataFrame(freqReview)
    dfName = []
    for c in topk:
        dfName.append(c[0])
    freqReviewDf.columns = dfName
    finalreviewDf = reviewDF.join(freqReviewDf)
    finaldf = hotelDf[['hotelName','ratingScore','groundTruth']].join(finalreviewDf)
    return topk, finaldf
```

In [0]:
```
# This grammar is described in the paper by S. N. Kim,
# T. Baldwin, and M.-Y. Kan.
# Evaluating n-gram based evaluation metrics for automatic
# keyphrase extraction.
# Technical report, University of Melbourne, Melbourne 2010.
grammar = r"""
    NBAR:
        {<NN.*|JJ>*<NN.*>}  # Nouns and Adjectives, terminated with Nouns

    NP:
        {<NBAR>}
        {<NBAR><IN><NBAR>}  # Above, connected with in/of/etc...
"""
```

In [0]:
```
# Part of Speech Tagging
# Google: https://en.wikipedia.org/wiki/Part-of-speech_tagging
tagger = PerceptronTagger()
pos_tag = tagger.tag
# Create phrase tree
chunker = nltk.RegexpParser(grammar)
# tree= chunker.parse(taggedToks)
```

In [0]:
```
topk_phrase, finaldf_phrase = newDataFrameTransformation(hotelDf, reviewDF, k=500)
```

In [32]:
```
topk_phrase
```

```
Out[32]: [('hot water', 68),
          ('room servic', 51),
          ('swim pool', 50),
          ('main road', 49),
          ('good place', 47),
          ('good view', 40),
          ('il palazzo', 39),
          ('nthe room', 38),
          ('good food', 36),
          ('delux room', 35),
          ('great place', 30),
          ('good experi', 30),
          ('valley view', 29),
          ('valley view room', 29),
          ('tabl land', 29),
          ('prospect hotel', 29),
          ('panchgani market', 28),
          ('great time', 27),
          ('main market', 27),
          ('hotel staff', 27),
          ('nthe staff', 26),
          ('hira baug', 26),
          ('long weekend', 25),
          ('food qualiti', 25),
          ('parsi point', 25),
          ('nice place', 24),
          ('nthe food', 23),
          ('first time', 23),
          ('nthe hotel', 23),
          ('great view', 22),
          ('next day', 22),
          ('good room', 21),
          ('hotel room', 21),
          ('modern amen', 21),
          ('super delux room', 21),
          ('good hotel', 20),
          ('old world charm', 20),
          ('first floor', 19),
          ('tabl tenni', 19),
          ('hotel mala', 18),
          ('beauti view', 17),
          ('beauti place', 17),
          ('hotel prospect', 17),
          ('ground floor', 16),
          ('non veg', 16),
          ('night stay', 16),
          ('nice view', 16),
          ('hustl bustl', 16),
          ('dine room', 16),
          ('extra bed', 15),
          ('famili friend', 15),
          ('larg group', 15),
          ('strawberri farm', 15),
          ('valley face room', 15),
          ('amaz view', 15),
          ('good staff', 15),
          ('approach road', 15),
          ('live room', 15),
          ('indoor game', 14),
          ('good thing', 14),
          ('great food', 14),
          ('good servic', 14),
          ('spaciou room', 14),
          ('mapro garden', 13),
          ('play area', 13),
          ('raini season', 13),
('mount view', 13),
('ravin hotel', 13),
('second visit', 13),
('overal experi', 12),
('good time', 12),
('earli morn', 12),
('parsi famili', 12),
('mount view heritag', 12),
('good care', 12),
('veg food', 12),
('clean room', 12),
('comfort stay', 11),
('bad experi', 11),
('good valu', 11),
('heritag hotel', 11),
('delici food', 11),
('peak season', 11),
('signatur crest', 11),
('recept area', 11),
('blue countri resort', 11),
('good locat', 11),
('breakfast lunch', 11),
('short stay', 10),
('small kid', 10),
('help staff', 10),
('qualiti time', 10),
('nthe place', 10),
('perfect place', 10),
('googl map', 10),
('heritag properti', 10),
('doubl bed', 10),
('good deal', 10),
('hotel manag', 10),
('warm welcom', 10),
('pool tabl', 10),
('top floor', 10),
('star hotel', 10),
('senior citizen', 10),
('magic show', 10),
('last minut', 10),
('log hous', 10),
('nice hotel', 10),
('bhatia villa', 10),
('courteou staff', 9),
('carrom board', 9),
('market area', 9),
('nthe resort', 9),
('littl bit', 9),
('great stay', 9),
('parsi food', 9),
('budget hotel', 9),
('hill station', 9),
('basic amen', 9),
('wonder place', 9),
('peac stay', 9),
('great experi', 9),
('panchgani mahabaleshwar', 9),
('peac place', 9),
('pleasant stay', 9),
('french window', 9),
('buffet breakfast', 9),
('breakfast buffet', 9),
('reason price', 9),
('millennium park', 9),
('last week', 9),
('camp fire', 9),
('front desk', 9),
('board game', 9),
('cold water', 8),
('big room', 8),
('next visit', 8),
('first experi', 8),
('citi life', 8),
('housekeep staff', 8),
('first thing', 8),
('game room', 8),
('big group', 8),
('nice experi', 8),
('staff member', 8),
('custom servic', 8),
('super delux', 8),
('ac room', 8),
('pool area', 8),
('wonder experi', 8),
('hotel millennium park', 8),
('beauti garden', 8),
('new year', 8),
('travel organis', 8),
('care taker', 7),
('excel place', 7),
('mani time', 7),
('summer plaza', 7),
('sofa cum bed', 7),
('lcd tv', 7),
('mani option', 7),
('bed sheet', 7),
('nice stay', 7),
('pure veg', 7),
('wonder stay', 7),
('standard room', 7),
('min walk', 7),
('authent parsi food', 7),
('scenic view', 7),
('long time', 7),
('nthe owner', 7),
('room size', 7),
('huge properti', 7),
('old properti', 7),
('spectacular view', 7),
('tt tabl', 7),
('high ceil', 7),
('air condition', 7),
('great servic', 7),
('good stay', 7),
('ground floor room', 7),
('awesom view', 7),
('good properti', 7),
('balconi room', 7),
('nthe properti', 7),
('recept staff', 7),
('garden area', 7),
('extra charg', 7),
('famili room', 7),
('amus park', 7),
('good work', 7),
('open space', 7),
('duplex room', 7),
('nice locat', 7),
('natur lover', 7),
('tenni court', 7),
('beauti valley view', 6),
('good resort', 6),
('prompt servic', 6),
('hous restaur', 6),
('execut room', 6),
('next time', 6),
('door lock', 6),
('main market area', 6),
('good restaur', 6),
('relax weekend', 6),
('hotel ravin', 6),
('vegetarian food', 6),
('awesom place', 6),
('reason rate', 6),
('mani hotel', 6),
('non vegetarian', 6),
('warm hospit', 6),
('home stay', 6),
('new room', 6),
('person attent', 6),
('ideal place', 6),
('food tast', 6),
('scenic beauti', 6),
('good option', 6),
('view room', 6),
('great valu', 6),
('small room', 6),
('good choic', 6),
('trip advisor', 6),
('restaur staff', 6),
('sign board', 6),
('premium suit', 6),
('amaz experi', 6),
('monsoon season', 6),
('co oper', 6),
('order food', 6),
('mountain view', 6),
('excel view', 6),
('room room', 6),
('sunset cottag', 6),
('help food', 6),
('amaz place', 6),
('comfort bed', 6),
('hotel mount view execut', 6),
('mount view execut', 6),
('terra camp', 6),
('premium room', 6),
('mr gaurav', 6),
('clean staff', 5),
('summer plaza resort', 5),
('averag experi', 5),
('amaz food', 5),
('bath tub', 5),
('hotel locat', 5),
('high expect', 5),
('basic thing', 5),
('great valley view', 5),
('il pallazo', 5),
('larg famili', 5),
('room food', 5),
('first trip', 5),
('krishna river', 5),
('wonder time', 5),
('fresh strawberri', 5),
('nthi place', 5),
('first impress', 5),
('fresh food', 5),
('fantast view', 5),
('beauti locat', 5),
('good review', 5),
('parsi cuisin', 5),
('servic staff', 5),
('limit option', 5),
('limit menu', 5),
('excel room', 5),
('famili stay', 5),
('relax stay', 5),
('memor experi', 5),
('size bed', 5),
('age group', 5),
('day stay', 5),
('tea coffe', 5),
('room key', 5),
('larg room', 5),
('excel servic', 5),
('wonder hospit', 5),
('room tariff', 5),
('memor stay', 5),
('good amen', 5),
('luxuri hotel', 5),
('good part', 5),
('power cut', 5),
('good qualiti', 5),
('hotel food', 5),
('big balconi', 5),
('breakfast spread', 5),
('extra mile', 5),
('great job', 5),
('nthe view', 5),
('dine area', 5),
('fresh air', 5),
('tree hous', 5),
('sydney point', 5),
('mani place', 5),
('floor room', 5),
('gorgeou view', 5),
('second time', 5),
('bedroom villa', 5),
('entir staff', 5),
('valley side', 5),
('strawberri field', 5),
('welcom drink', 5),
('water park', 5),
('nthe locat', 4),
('weekend getaway', 4),
('main town', 4),
('aloo paratha', 4),
('jain food', 4),
('panchgani main market', 4),
('indoor swim pool', 4),
('make sure', 4),
('food option', 4),
('special mention', 4),
('jain templ', 4),
('weekend stay', 4),
('non veg food', 4),
('non vegetarian food', 4),
('old peopl', 4),
('whole resort', 4),
('room rate', 4),
('prochi cottag', 4),
('nthi properti', 4),
('love famili', 4),
('main build', 4),
('day trip', 4),
('authent parsi cuisin', 4),
('mani guest', 4),
('spend time', 4),
('panoram view', 4),
('yummi food', 4),
('owner famili', 4),
('clean washroom', 4),
('lawn area', 4),
('free breakfast', 4),
('room number', 4),
('person touch', 4),
('bed room', 4),
('first room', 4),
('right place', 4),
('addit charg', 4),
('panchgani town', 4),
('sever time', 4),
('excel food', 4),
('polit help', 4),
('electr kettl', 4),
('loud music', 4),
('hotel view', 4),
('short trip', 4),
('conveni locat', 4),
('hotel rate', 4),
('holiday home', 4),
('mr parth', 4),
('first night', 4),
('absolut valu', 4),
('money food', 4),
('earli check', 4),
('rainforest restaur', 4),
('small balconi', 4),
('market place', 4),
('next trip', 4),
('blue countri', 4),
('good size', 4),
('front desk staff', 4),
('ampl space', 4),
('nthe pool', 4),
('regular visitor', 4),
('friendli staff', 4),
('nice garden', 4),
('good ncon', 4),
('big properti', 4),
('park space', 4),
('cant comment', 4),
('nice swim pool', 4),
('good varieti', 4),
('last moment', 4),
('mini theatr', 4),
('nfood qualiti', 4),
('quiet place', 4),
('room rent', 4),
('nthe servic', 4),
('tent room', 4),
('cleartrip com', 4),
('beauti properti', 4),
('nthe restaur', 4),
('bad condit', 4),
('food item', 4),
('sport room', 4),
('luxuri tent', 4),
('love time', 4),
('long drive', 4),
('resort manag', 4),
('entir valley', 4),
('wi fi', 4),
('love view', 4),
('amaz locat', 4),
('citi center', 4),
('natur beauti', 4),
('modern facil', 4),
('main hous', 4),
('entir day', 4),
('par excel', 4),
('suit room', 4),
('queen size', 4),
('car park', 4),
('good budget hotel', 4),
('tourist spot', 4),
('buffet meal', 4),
('hous guest', 4),
('hotel owner', 4),
('perfect getaway', 4),
('entir stay', 4),
('bhillar villag', 4),
('awesom hotel', 4),
('mumbai pune expressway', 4),
('wonder view', 4),
('plu point', 4),
('complimentari breakfast', 4),
('exot home stay', 4),
('wonder properti', 4),
('hotel resid', 4),
('nroom servic', 4),
('chine food', 4),
('help room', 4),
('theme park', 4),
('horror hous', 4),
('hotel valley nest', 4),
('play room', 3),
('free wi fi', 3),
('simpl thing', 3),
('attach balconi', 3),
('room clean', 3),
('min drive', 3),
('tabl top mountain', 3),
('short staf', 3),
('spa facil', 3),
('singl day', 3),
('singl time', 3),
('pleasant place', 3),
('maintain staff', 3),
('good meal', 3),
('comfort room', 3),
('major meal', 3),
('hotel area', 3),
('complet packag', 3),
('book room', 3),
('ur room', 3),
('second day', 3),
('neg point', 3),
('indian cuisin', 3),
('bad smell', 3),
('checkout time', 3),
('resort staff', 3),
('mani pictur', 3),
('luxuri amen', 3),
('pure vegetarian resort', 3),
('decent food', 3),
 ('mani thing', 3),
 ('room bathroom', 3),
 ('person interest', 3),
 ('minut walk', 3),
 ('friend famili', 3),
 ('panchgani u002f mahabaleshwar', 3),
 ('hot tea', 3),
 ('perfect locat', 3),
 ('big thank', 3),
 ('horribl place', 3),
 ('mount view hotel', 3),
 ('charm food', 3),
 ('hot water facil', 3),
 ('quaint place', 3),
 ('dhom dam', 3),
 ('lush greeneri', 3),
 ('tea u002f coffe', 3),
 ('mani year', 3),
 ('beauti hotel', 3),
 ('warm servic', 3),
 ('peac properti', 3),
 ('great deal', 3),
 ('decent place', 3),
 ('next room', 3),
 ('fix menu', 3),
 ('good wifi', 3),
 ('heritag look', 3),
 ('huge room', 3),
 ('multipl time', 3),
 ('fond memori', 3),
 ('hospit industri', 3),
 ('high point', 3),
 ('2nd time', 3),
 ('travel plan', 3),
 ('common area', 3),
 ('hotel ground', 3),
 ('uniqu experi', 3),
 ('old hotel', 3)]
```

In [0]:

```
# Revise the previous dataframe transform function...
def getTopKPhrase(df, k, label_value, label_column='groundTruth', operation=operator.eq, value_
column='reviewCol'):
    counter = Counter()
    for review in df.loc[operation(df[label_column],label_value)][value_column]:
            counter.update(flatten([word
                            for word
                            in get_terms(chunker.parse(pos_tag(re.findall(r'\w+', review))))
                            ]))
    topk = counter.most_common(k)
    return topk
```

```
In [34]: topkPhraseGroundPos = getTopKPhrase(df=itemAnalysisDf, k=50, label_value='positive')
         topkPhraseGroundPos
```

```
Out[34]: [('il palazzo', 38),
          ('swim pool', 36),
          ('main road', 35),
          ('good place', 34),
          ('hot water', 31),
          ('good food', 31),
          ('nthe room', 29),
          ('prospect hotel', 29),
          ('great place', 28),
          ('good experi', 27),
          ('great time', 26),
          ('main market', 26),
          ('good view', 26),
          ('tabl land', 26),
          ('hira baug', 25),
          ('room servic', 24),
          ('panchgani market', 24),
          ('nthe staff', 23),
          ('nice place', 21),
          ('delux room', 21),
          ('parsi point', 21),
          ('valley view', 20),
          ('valley view room', 20),
          ('modern amen', 20),
          ('nthe food', 19),
          ('old world charm', 19),
          ('first time', 17),
          ('beauti place', 17),
          ('long weekend', 17),
          ('hotel prospect', 17),
          ('nthe hotel', 16),
          ('hotel room', 16),
          ('beauti view', 15),
          ('non veg', 15),
          ('famili friend', 15),
          ('hustl bustl', 15),
          ('strawberri farm', 15),
          ('good room', 14),
          ('great view', 14),
          ('nice view', 14),
          ('food qualiti', 14),
          ('spaciou room', 14),
          ('good staff', 14),
          ('super delux room', 14),
          ('amaz view', 14),
          ('live room', 14),
          ('hotel mala', 14),
          ('indoor game', 13),
          ('great food', 13),
          ('hotel staff', 13)]
```

```
In [35]: topkPhraseGroundNeg = getTopKPhrase(df=itemAnalysisDf, k=50, label_value='negative')
         topkPhraseGroundNeg
```

```
Out[35]: [('hot water', 37),
          ('room servic', 27),
          ('swim pool', 14),
          ('next day', 14),
          ('good view', 14),
          ('delux room', 14),
          ('main road', 14),
          ('hotel staff', 14),
          ('good place', 13),
          ('food qualiti', 11),
          ('good thing', 10),
          ('bad experi', 10),
          ('good hotel', 10),
          ('tabl tenni', 9),
          ('valley view', 9),
          ('nthe room', 9),
          ('valley view room', 9),
          ('long weekend', 8),
          ('great view', 8),
          ('peak season', 8),
          ('travel organis', 8),
          ('first floor', 7),
          ('bed sheet', 7),
          ('good room', 7),
          ('nthe hotel', 7),
          ('super delux room', 7),
          ('play area', 6),
          ('first time', 6),
          ('air condition', 6),
          ('approach road', 6),
          ('carrom board', 5),
          ('door lock', 5),
          ('hotel room', 5),
          ('extra bed', 5),
          ('hotel manag', 5),
          ('small room', 5),
          ('good food', 5),
          ('hill station', 5),
          ('recept area', 5),
          ('premium suit', 5),
          ('good locat', 5),
          ('ravin hotel', 5),
          ('ground floor', 4),
          ('averag experi', 4),
          ('bath tub', 4),
          ('new room', 4),
          ('game room', 4),
          ('tt tabl', 4),
          ('night stay', 4),
          ('holiday home', 4)]
```

Answer 2(b):

- 'il plazzo' had highest frequency in positive reviews. Il Plazzo is in iteself a hotel name and it had maximum 100 reviews. Thus, term frequency based approach is biased towards/ dominated by the hotels with high reviews.
- Similarily, 'swimmming pool' appeared frequently in positive reviews. This may be due to the where in one of the hotel with the highest review has a good swimming pool. However, this cannote be generalized to the city.
- Panchgini has a very good 'table land', 'parsi point', 'panchgini market' and 'hira baug' for tourists
- The place has good 'strawberry farms'
- Even though 'hot water appeared frequently in negative reviews, it also appeard almsot equally in the positive reviews. Thus, it cannot be concluded that hot water isn't good in the city. Same goes for the phrase 'room servic'

Q3 Mutual Information

Q3 (a)

```
In [0]: # get Top K mutual information terms from the dataframe
        def getMI(topk, df, label_column='groundTruth'):
            miScore = []
            for word in topk:
                miScore.append([word[0]]+[metrics.mutual_info_score(df[label_column], df[word[0]])])
            miScoredf = pandas.DataFrame(miScore).sort_values(1,ascending=0)
            miScoredf.columns = ['Word','MI Score']
            return miScoredf
```

```
In [0]: topk, finaldf = dataFrameTransformation(hotelDf, reviewDF, k=1500)
```

```
In [38]: len(topk)
```

```
Out[38]: 1500
```

```
In [39]: miScoredf = getMI(topk, finaldf)
         miScoredf.head(50)
```

Out[39]:

	Word	MI Score
226	dirty	0.040236
121	bad	0.037281
276	poor	0.035465
36	water	0.031018
504	worst	0.025540
590	horrible	0.023645
3	room	0.023550
286	working	0.022306
540	pathetic	0.020365
80	bathroom	0.019084
525	broken	0.017826
680	rude	0.017706
43	beautiful	0.016902
199	average	0.015717
90	awesome	0.014939
973	badly	0.014785
378	charged	0.014530
53	night	0.014501
213	asked	0.013860
292	charge	0.013215
123	perfect	0.012734
38	even	0.012719
24	best	0.012067
772	sheets	0.011681
64	home	0.011026
1051	intercom	0.010905
920	terrible	0.010905
132	lovely	0.010775
62	booked	0.010536
42	excellent	0.010312
104	reception	0.010087
261	towels	0.009896
418	looks	0.009817
75	parsi	0.009794
150	delicious	0.009752
357	dont	0.009648
87	bed	0.009613

	Word	MI Score
318	call	0.009557
1394	unprofessional	0.009528
170	loved	0.009523
231	per	0.009435
74	better	0.009428
256	told	0.009213
40	small	0.009059
55	amazing	0.008842
84	money	0.008827
469	condition	0.008795
871	confirmed	0.008656
441	pay	0.008574
105	enjoyed	0.008493

Answer 3(a):

- Top 500 tokens cannot be used becuase there can be tokens with lower frequency but higher MI. Thus, finding threshold of the k such that the rank of top 50 MI remains the same
- Negative terms had higher MI compared to the positive i.e a sign of presence of the negative terms in negative reviews as well as its absence in positive reviews
- There is absense of local-specific terms such as 'valley', 'mountains', 'parsi' which appeared in the frequency based approach. This may be due to the case that such terms are not able to discren well between the positive or negative review i.e. these terms have their presence in positive and well as negative reviews

Q3 (b)

```
In [0]: topk_phrase, finaldf_phrase = newDataFrameTransformation(hotelDf, reviewDF, k=5000)
```

```
In [41]: len(topk_phrase)
```

```
Out[41]: 5000
```

In [42]:
```
miScoredf_phrase = getMI(topk_phrase, finaldf_phrase)
miScoredf_phrase.head(50)
```

Out[42]:

	Word	MI Score
161	bed sheet	0.006047
0	hot water	0.005557
1	room servic	0.005304
78	bad experi	0.004372
59	good thing	0.004205
15	prospect hotel	0.003646
82	peak season	0.003479
430	horror hous	0.003448
222	small room	0.002829
41	beauti place	0.002773
6	il palazzo	0.002755
18	main market	0.002604
17	great time	0.002604
50	famili friend	0.002599
52	strawberri farm	0.002599
429	theme park	0.002584
455	bad smell	0.002584
499	old hotel	0.002584
549	horribl experi	0.002584
662	proper place	0.002584
471	horribl place	0.002584
566	map plan	0.002584
582	bad servic	0.002584
561	ok servic	0.002584
496	common area	0.002584
554	clean toilet	0.002584
492	hospit industri	0.002584
490	multipl time	0.002584
19	hotel staff	0.002433
62	spaciou room	0.002425
248	bath tub	0.002088
202	door lock	0.002088
278	room key	0.002088
10	great place	0.002077
72	parsi famili	0.002077
80	heritag hotel	0.001903
33	modern amen	0.001863

	Word	MI Score
9	delux room	0.001756
73	mount view heritag	0.001729
90	help staff	0.001729
91	qualiti time	0.001729
1345	rude staff	0.001722
1351	expens price	0.001722
1346	power backup	0.001722
1156	restaur servic	0.001722
1047	basic facil	0.001722
922	store room	0.001722
704	jet spray	0.001722
1043	season rate	0.001722
929	last night	0.001722

Answer 3(b):

- Top 500 tokens of phrases cannot be used becuase there can be tokens with lower frequency but higher MI. Thus, finding threshold of the k such that the rank of top 50 MI remains the same
- The phrases such as 'bed sheets', 'hot water', 'room servic' with highest MI does not tell whether it was used in a postive or negative context. This is due to the fact that MI is calculated over all the possible combinations of ground truth as well as the presense/absence of a phrase

Q4 Pointwise Mutual Information

Q4 (a)

In [0]:
```
# Simple example of getting pairwise mutual information of a term
def pmiCal(df, x):
    pmilist=[]
    for i in ['positive','negative']:
        for j in [0,1]:
            px = sum(df['groundTruth']==i)/len(df) #FRAC OF REVIEWS POS OR NEG IN DF
            py = sum(df[x]==j)/len(df) #FRAC OF REVIEWS WITH WORD x
            pxy = len(df[(df['groundTruth']==i) & (df[x]==j)])/len(df)
            if px ==0 or py == 0:
              pmi= -99
            elif pxy==0:#Log 0 cannot happen
                pmi = math.log((pxy+0.0001)/(px*py))
            else:
                pmi = math.log(pxy/(px*py))
            pmilist.append([i]+[j]+[px]+[py]+[pxy]+[pmi])
    pmidf = pandas.DataFrame(pmilist)
    pmidf.columns = ['x','y','px','py','pxy','pmi']
    return pmidf
```

```
In [0]: def pmiIndivCal(df,x,gt, label_column='groundTruth'):
            px = sum(df[label_column]==gt)/len(df)
            py = sum(df[x]==1)/len(df)
            pxy = len(df[(df[label_column]==gt) & (df[x]==1)])/len(df)
            if px ==0 or py == 0:
                pmi= -99
            elif pxy==0:#Log 0 cannot happen
                pmi = math.log((pxy+0.0001)/(px*py))
            else:
                pmi = math.log(pxy/(px*py))
            return pmi
```

```
In [0]: # Compute PMI for all terms and all possible labels
        def pmiForAllCal(df, label_column='groundTruth', topk=topk):
            #Try calculate all the pmi for top k and store them into one pmidf dataframe
            pmilist = []
            pmiposlist = []
            pmineglist = []
            for word in tqdm(topk):
                pmilist.append([word[0]]+[pmiCal(df,word[0])])
                pmiposlist.append([word[0]]+[pmiIndivCal(df,word[0],'positive',label_column)])
                pmineglist.append([word[0]]+[pmiIndivCal(df,word[0],'negative',label_column)])
            pmidf = pandas.DataFrame(pmilist)
            pmiposlist = pandas.DataFrame(pmiposlist)
            pmineglist = pandas.DataFrame(pmineglist)
            pmiposlist.columns = ['word','pmi']
            pmineglist.columns = ['word','pmi']
            pmidf.columns = ['word','pmi']
            return pmiposlist, pmineglist, pmidf
```

```
In [0]: topk, finaldf = dataFrameTransformation(hotelDf, reviewDF, k=1500)
```

```
In [47]: len(topk)

Out[47]: 1500
```

```
In [48]: pmiposlist, pmineglist, pmidf = pmiForAllCal(finaldf)

         100%|██████████| 1500/1500 [00:38<00:00, 39.10it/s]
```

```
In [49]: pmiposlist.sort_values(by='pmi', ascending=0).head(50)
```

```
Out[49]:
```

	word	pmi
1094	log	0.281581
1251	heaven	0.281581
517	truly	0.281581
1356	skeptical	0.281581
921	outstanding	0.281581
397	faram	0.281581
1370	delicacies	0.281581
1395	impressed	0.281581
244	prospect	0.281581
1278	stroll	0.281581
1361	cooking	0.281581
1192	era	0.281581
638	host	0.281581
894	humble	0.281581
1140	thoroughly	0.281581
975	sun	0.281581
942	hosts	0.281581
1466	ensured	0.281581
1327	ud83c	0.281581
1463	parsee	0.281581
1045	owned	0.281581
1029	decision	0.281581
1069	bhilar	0.281581
1417	bike	0.281581
1457	treat	0.281581
868	bhatia	0.281581
1059	arrangements	0.281581
719	sightseeing	0.281581
859	cake	0.281581
1169	udc4d	0.281581
1345	discotheque	0.281581
1480	appointed	0.281581
1107	fabulous	0.281581
946	bedrooms	0.281581
902	beautifully	0.281581
1496	u002fc	0.281581
1391	ngreat	0.281581

	word	pmi
1247	prashant	0.281581
903	warmth	0.281581
986	masala	0.281581
1012	birthday	0.281581
1211	antique	0.281581
1404	conference	0.281581
1214	wide	0.281581
1280	veerzad	0.281581
1285	gem	0.281581
485	ud83d	0.281581
308	homely	0.264189
387	special	0.259602
251	palazzo	0.257483

In [50]:
```
pmineglist.sort_values(by='pmi', ascending=0).head(50)
```

Out[50]:

	word	pmi
871	confirmed	1.404825
1402	oyo	1.404825
790	u003e	1.404825
1394	unprofessional	1.404825
590	horrible	1.404825
973	badly	1.404825
1448	pillow	1.404825
504	worst	1.374053
680	rude	1.362265
920	terrible	1.340286
1051	intercom	1.340286
540	pathetic	1.335832
525	broken	1.327864
226	dirty	1.319303
1299	waste	1.317813
1171	refused	1.309515
1397	stinking	1.299464
276	poor	1.295626
772	sheets	1.258221
1066	understand	1.250674
1161	drinking	1.237771
1246	refund	1.222503
1259	nwhen	1.204154
1398	walls	1.204154
802	smell	1.181681
1198	damp	1.181681
1368	towel	1.181681
1468	tiny	1.181681
1326	chargeable	1.181681
1461	sad	1.181681
950	saying	1.163663
286	working	1.158692
1301	agreed	1.153510
1447	shifted	1.153510
1451	plates	1.153510
1208	cons	1.142461
966	lock	1.142461

	word	pmi
1152	mosquitoes	1.142461
937	arrived	1.136561
999	closed	1.136561
1329	bill	1.117143
1138	insects	1.117143
1342	assured	1.117143
121	bad	1.110585
1080	neither	1.094670
1244	remote	1.086371
1409	manage	1.086371
787	okay	1.086371
1135	unless	1.068353
1109	using	1.068353

Answer 4(a):

- Unlike MI, in PMI, it is feasible to check if the token is used positively or negatively in the the review
- The presence of the terms like 'insects', 'mosquitoes' indicate that the place is prone to insect bites thereby leading to a negavtive customer experience
- The local-specific detail obtained in the positive PMI terms was negligible. However, there were names found in the positive PMI terms most likely indicating that the host's name

Q4 (b)

```
In [0]: topk_phrase, finaldf_phrase = newDataFrameTransformation(hotelDf, reviewDF, k=100)
```

```
In [52]: len(topk_phrase)
Out[52]: 100
```

```
In [53]: pmiposlist_phrase, pmineglist_phrase, pmidf_phrase = pmiForAllCal(finaldf_phrase,'groundTruth',
         topk_phrase)

         100%|██████████| 100/100 [00:01<00:00, 57.04it/s]
```

```
In [54]: pmiposlist_phrase.sort_values(by='pmi', ascending=0).head(50)
```

Out[54]:

	word	pmi
80	heritag hotel	0.281581
50	famili friend	0.281581
95	heritag properti	0.281581
41	beauti place	0.281581
91	qualiti time	0.281581
90	help staff	0.281581
42	hotel prospect	0.281581
52	strawberri farm	0.281581
62	spaciou room	0.281581
73	mount view heritag	0.281581
72	parsi famili	0.281581
15	prospect hotel	0.281581
6	il palazzo	0.243840
18	main market	0.242360
17	great time	0.242360
33	modern amen	0.232791
36	old world charm	0.230287
47	hustl bustl	0.217042
44	non veg	0.217042
10	great place	0.212588
21	hira baug	0.212588
55	good staff	0.212588
54	amaz view	0.212588
61	good servic	0.207473
60	great food	0.207473
58	indoor game	0.207473
68	second visit	0.201538
57	live room	0.194569
66	mount view	0.194569
71	earli morn	0.186271
79	good valu	0.186271
81	delici food	0.186271
93	perfect place	0.176220
99	warm welcom	0.176220
11	good experi	0.176220
94	googl map	0.163798
97	good deal	0.163798

	word	pmi
20	nthe staff	0.158978
14	tabl land	0.158978
40	beauti view	0.156418
46	nice view	0.148049
25	nice place	0.148049
8	good food	0.132049
16	panchgani market	0.127430
51	larg group	0.114527
63	mapro garden	0.114527
76	clean room	0.099259
24	parsi point	0.090526
26	nthe food	0.090526
39	hotel mala	0.087425

```
In [55]: pmineglist_phrase.sort_values(by='pmi', ascending=0).head(50)
```

Out[55]:

	word	pmi
78	bad experi	1.271293
82	peak season	1.086371
59	good thing	1.068353
1	room servic	0.752500
19	hotel staff	0.711678
98	hotel manag	0.711678
0	hot water	0.711678
30	next day	0.711678
35	good hotel	0.657610
38	tabl tenni	0.657610
64	play area	0.631635
86	good locat	0.616367
84	recept area	0.616367
23	food qualiti	0.583844
9	delux room	0.578146
56	approach road	0.488534
37	first floor	0.460363
67	ravin hotel	0.393224
29	great view	0.355003
5	good view	0.306213
75	veg food	0.306213
96	doubl bed	0.306213
49	extra bed	0.306213
31	good room	0.306213
22	long weekend	0.265391
12	valley view	0.234754
65	raini season	0.226170
13	valley view room	0.226170
28	nthe hotel	0.215241
2	swim pool	0.172681
3	main road	0.141133
69	overal experi	0.105542
45	night stay	0.083069
27	first time	0.061090
4	good place	0.018530
70	good time	0.018530
48	dine room	0.018530

	word	pmi
43	ground floor	0.018530
32	hotel room	-0.030260
7	nthe room	-0.035537
34	super delux room	-0.099253
88	short stay	-0.099253
53	valley face room	-0.204613
83	signatur crest	-0.204613
89	small kid	-0.204613
85	blue countri resort	-0.204613
92	nthe place	-0.204613
87	breakfast lunch	-0.299923
74	good care	-0.299923
77	comfort stay	-0.299923

Answer 4(b):

- The k in topkPhrase was tuned such that the resultant PMIs in top 50 are as distinct as possible and to include least negative PMIs
- 'strawberri farm', 'panchgini market', 'hira baugh', 'mapro gardern' are the local-specific places with postivie reviews. In fact, it is only PMI that the most renowned place 'mapro garden' was obtained. In general, reviews of the place are positive with good views
- People had negative reviews associated with rainy season, indicating monsoon may not be the best time to visit this place.
- Overall, people were not happy with the 'hotel staff', 'hotel water'. However, this can he hotel-specific and may not be implied for the city.

Q4 (c)

```
In [0]: topk_phrase, finaldf_phrase = newDataFrameTransformation(hotelDf, reviewDF, k=5000)
```

```
In [57]: hotelRatingDfGt['hotelName'].values[0]

Out[57]: 'Basilica Rediscover Serenity'
```

In [58]:
```
finaldf_phrase.loc[finaldf_phrase['hotelName']==hotelRatingDfGt['hotelName'].values[0]]
```

Out[58]:

	hotelName	ratingScore	groundTruth	reviewCol	vader	hot water	room servic	swim pool	main road	good place	good view	palazz
877	Basilica Rediscover Serenity	5	positive	"Best hotel to stay in the area with perfect l...	0.9732	0	0	0	0	0	0	0
878	Basilica Rediscover Serenity	5	positive	"Great modern rooms, amazing service at a good...	0.9761	0	0	0	0	0	0	0
879	Basilica Rediscover Serenity	5	positive	"One of the best resorts in Panchgani!!! We ha...	0.9893	0	0	0	0	0	0	0
880	Basilica Rediscover Serenity	5	positive	"Had stayed in this resort exclusively to prop...	0.4588	0	0	0	0	0	0	0
881	Basilica Rediscover Serenity	5	positive	"When you plan your visit to Panchgani or Maha...	0.9605	0	0	0	0	0	0	0

5 rows × 5005 columns

In [59]:
```
pmiposlist_phrase_top, pmineglist_phrase_top, pmidf_phrase_top = pmiForAllCal(finaldf_phrase.lo
c[finaldf_phrase['hotelName']==hotelRatingDfGt['hotelName'].values[0]],'groundTruth',topk_phras
e)
```

```
100%|██████████| 5000/5000 [01:06<00:00, 75.07it/s]
```

```
In [60]: pmiposlist_phrase_top.sort_values(by='pmi', ascending=0).head(50)
```

Out[60]:

	word	pmi
128	reason price	0.0
596	memor one	0.0
449	complet packag	0.0
415	perfect getaway	0.0
650	amaz hospit	0.0
759	flat tyre	0.0
622	special thank	0.0
1220	person care	0.0
271	famili stay	0.0
1256	mr mateen	0.0
81	delici food	0.0
120	peac stay	0.0
1257	delici staff	0.0
469	perfect locat	0.0
3335	good tri local dish	-99.0
3338	superior delux valley face room	-99.0
3328	friendli guy	-99.0
3337	point room servic	-99.0
3336	travel desk	-99.0
3330	good featur	-99.0
3334	sure everyon	-99.0
3333	morn person attent	-99.0
3332	lawn nfresh food nprompt servic	-99.0
3329	panchgani cool breez valley view open lawn minut	-99.0
3340	gener backup	-99.0
3331	huge properti nclean pool	-99.0
3339	clean nroom	-99.0
3344	horribl hotel unnecessarili	-99.0
3341	ac ngood food	-99.0
3351	pathet food	-99.0
3358	amaz thing	-99.0
3357	photo graph	-99.0
3356	pool accessori	-99.0
3355	tv recept poor bathroom water clog n1st day	-99.0
3354	corridor fridg	-99.0
3353	secur nbathroom window	-99.0
3352	mainten ni	-99.0

	word	pmi
3350	nsriniva popuri	-99.0
3342	sumptuou spread	-99.0
3349	trust ni	-99.0
3348	peopl cleartrip	-99.0
3347	elit custom	-99.0
3346	innoc travel	-99.0
3345	hotel peopl	-99.0
3326	nour kid	-99.0
3343	buffet breakfast nswim pool	-99.0
3327	parti game	-99.0
0	hot water	-99.0
3325	danc step	-99.0
3298	pleasant memori thing	-99.0

```
In [61]: pmineglist_phrase_top.sort_values(by='pmi', ascending=0).head(50)
```

Out[61]:

	word	pmi
0	hot water	-99
3330	good featur	-99
3337	point room servic	-99
3336	travel desk	-99
3335	good tri local dish	-99
3334	sure everyon	-99
3333	morn person attent	-99
3332	lawn nfresh food nprompt servic	-99
3331	huge properti nclean pool	-99
3329	panchgani cool breez valley view open lawn minut	-99
3339	clean nroom	-99
3328	friendli guy	-99
3327	parti game	-99
3326	nour kid	-99
3325	danc step	-99
3324	dj music	-99
3323	memor experi nwe	-99
3322	light music	-99
3338	superior delux valley face room	-99
3340	gener backup	-99
3437	import thing food	-99
3350	nsriniva popuri	-99
3357	photo graph	-99
3356	pool accessori	-99
3355	tv recept poor bathroom water clog n1st day	-99
3354	corridor fridg	-99
3353	secur nbathroom window	-99
3352	mainten ni	-99
3351	pathet food	-99
3349	trust ni	-99
3341	ac ngood food	-99
3348	peopl cleartrip	-99
3347	elit custom	-99
3346	innoc travel	-99
3345	hotel peopl	-99
3344	horribl hotel unnecessarili	-99
3343	buffet breakfast nswim pool	-99

	word	pmi
3342	sumptuou spread	-99
3321	restaur manag mr nikhil	-99
3320	full occup	-99
3319	amaz clean	-99
3291	ampl place	-99
3298	pleasant memori thing	-99
3297	decent wat	-99
3296	huge resort	-99
3295	suffici park	-99
3294	good swim pool	-99
3293	big park space nice restaur tabl land	-99
3292	good lawn nice plantat	-99
3290	nice discount	-99

```
In [62]: len(topk)

Out[62]: 1500
```

```
In [63]: pmiposlist_top, pmineglist_top, pmidf_top = pmiForAllCal(finaldf.loc[finaldf['hotelName']==hote
         lRatingDfGt['hotelName'].values[0]],'groundTruth',topk)

         100%|██████████| 1500/1500 [00:19<00:00, 78.34it/s]
```

```
 In [0]: df1=finaldf.loc[finaldf['hotelName']==hotelRatingDfGt['hotelName'].values[0]]
```

```
In [65]: pmiCal(df1, 'hotel')

Out[65]:
```

	x	y	px	py	pxy	pmi
0	positive	0	1.0	0.6	0.6	0.0
1	positive	1	1.0	0.4	0.4	0.0
2	negative	0	0.0	0.6	0.0	-99.0
3	negative	1	0.0	0.4	0.0	-99.0

```
In [66]: pmiposlist_top.sort_values(by='pmi', ascending=0).head(50)
```

```
Out[66]:
```

	word	pmi
0	hotel	0.0
127	needs	0.0
110	hospitality	0.0
111	worth	0.0
118	want	0.0
123	perfect	0.0
361	thanks	0.0
598	ngood	0.0
709	attention	0.0
150	delicious	0.0
354	nrooms	0.0
136	care	0.0
1058	exceptional	0.0
348	came	0.0
1219	behaviour	0.0
346	vacation	0.0
723	light	0.0
373	personal	0.0
375	reasonable	0.0
859	cake	0.0
98	however	0.0
97	owner	0.0
94	make	0.0
383	making	0.0
736	game	0.0
847	nservice	0.0
387	special	0.0
86	dinner	0.0
83	every	0.0
78	friendly	0.0
77	recommend	0.0
1	good	0.0
1455	uncle	0.0
146	sure	0.0
151	always	0.0
70	mahabaleshwar	0.0
282	memorable	0.0

	word	pmi
299	indoor	0.0
205	given	0.0
296	recommended	0.0
1014	secluded	0.0
570	equipped	0.0
285	modern	0.0
1275	flat	0.0
529	complete	0.0
225	kind	0.0
233	children	0.0
265	called	0.0
609	takes	0.0
1297	properties	0.0

```
In [67]: pmineglist_top.sort_values(by='pmi', ascending=0).head(50)
```

```
Out[67]:
```

	word	pmi
0	hotel	-99
997	helping	-99
1006	inr	-99
1005	drinks	-99
1004	carry	-99
1003	evenings	-99
1002	present	-99
1001	behind	-99
1000	august	-99
999	closed	-99
998	atleast	-99
996	whether	-99
985	blankets	-99
995	nwould	-99
994	plenty	-99
993	corner	-99
992	hand	-99
991	true	-99
990	veranda	-99
989	school	-99
988	wooden	-99
987	baby	-99
1007	pls	-99
1008	arrange	-99
1009	july	-99
1010	chef	-99
1029	decision	-99
1028	requested	-99
1027	reaching	-99
1026	satisfied	-99
1025	problems	-99
1024	advantage	-99
1023	move	-99
1022	december	-99
1021	bunglow	-99
1020	electricity	-99
1019	dec	-99

	word	pmi
1018	krishna	-99
1017	wall	-99
1016	soft	-99
1015	valleys	-99
1014	secluded	-99
1013	terra	-99
1012	birthday	-99
1011	shop	-99
986	masala	-99
984	husband	-99
1031	stairs	-99
950	saying	-99
959	security	-99

```
In [68]: hotelRatingDfGt.tail()
```

```
Out[68]:
```

	hotelName	avgRatingScore
51	Hotel Residency	2.586207
37	The Dhanhills	2.550000
56	Alliance Tents and Accommodations	1.750000
17	OYO 9566 Hotel JK Excellency	1.400000
2	Ashirwad Bungalow	1.000000

```
In [69]: finaldf_phrase.loc[finaldf_phrase['hotelName']==hotelRatingDfGt.tail(2)['hotelName'].values[0]]
```

Out[69]:

	hotelName	ratingScore	groundTruth	reviewCol	vader	hot water	room servic	swim pool	main road	good place	good view	i palazzo
696	OYO 9566 Hotel JK Excellency	1	negative	"We are senior citizens and my wife is diabeti...	-0.8838	0	0	0	0	0	0	0
697	OYO 9566 Hotel JK Excellency	1	negative	"No DG back, staffs very arrogant, worst break...	-0.7623	0	0	0	0	0	0	0
698	OYO 9566 Hotel JK Excellency	3	negative	"We stayed there for a day as we were heading ...	0.8689	0	0	0	0	0	0	0
699	OYO 9566 Hotel JK Excellency	1	negative	"Terrible stay terrible I must say.\nNot recom...	-0.8860	0	0	0	0	0	0	0
700	OYO 9566 Hotel JK Excellency	1	negative	"We (me and my friend) had been there for one ...	-0.9587	0	0	0	0	0	0	0

5 rows × 5005 columns

In [70]:
```
pmiposlist_phrase_bot, pmineglist_phrase_bot, pmidf_phrase_bot = pmiForAllCal(finaldf_phrase.lo
c[finaldf_phrase['hotelName']==hotelRatingDfGt.tail(2)['hotelName'].values[0]],'groundTruth',to
pk_phrase)
pmiposlist_phrase_bot.sort_values(by='pmi', ascending=0).head(50)
```

```
100%|██████████| 5000/5000 [01:06<00:00, 75.59it/s]
```

```
Out[70]:
```

	word	pmi
0	hot water	-99
3330	good featur	-99
3337	point room servic	-99
3336	travel desk	-99
3335	good tri local dish	-99
3334	sure everyon	-99
3333	morn person attent	-99
3332	lawn nfresh food nprompt servic	-99
3331	huge properti nclean pool	-99
3329	panchgani cool breez valley view open lawn minut	-99
3339	clean nroom	-99
3328	friendli guy	-99
3327	parti game	-99
3326	nour kid	-99
3325	danc step	-99
3324	dj music	-99
3323	memor experi nwe	-99
3322	light music	-99
3338	superior delux valley face room	-99
3340	gener backup	-99
3437	import thing food	-99
3350	nsriniva popuri	-99
3357	photo graph	-99
3356	pool accessori	-99
3355	tv recept poor bathroom water clog n1st day	-99
3354	corridor fridg	-99
3353	secur nbathroom window	-99
3352	mainten ni	-99
3351	pathet food	-99
3349	trust ni	-99
3341	ac ngood food	-99
3348	peopl cleartrip	-99
3347	elit custom	-99
3346	innoc travel	-99
3345	hotel peopl	-99
3344	horribl hotel unnecessarili	-99
3343	buffet breakfast nswim pool	-99

	word	pmi
3342	sumptuou spread	-99
3321	restaur manag mr nikhil	-99
3320	full occup	-99
3319	amaz clean	-99
3291	ampl place	-99
3298	pleasant memori thing	-99
3297	decent wat	-99
3296	huge resort	-99
3295	suffici park	-99
3294	good swim pool	-99
3293	big park space nice restaur tabl land	-99
3292	good lawn nice plantat	-99
3290	nice discount	-99

```
In [71]: pmineglist_phrase_bot.sort_values(by='pmi', ascending=0).head(50)
```

Out[71]:

	word	pmi
1175	continu knock	0.0
19	hotel staff	0.0
1176	seclud place	0.0
117	hill station	0.0
30	next day	0.0
915	hotel nwe	0.0
137	first experi	0.0
103	senior citizen	0.0
123	peac place	0.0
398	wi fi	0.0
1174	sugar level	0.0
3356	pool accessori	-99.0
3335	good tri local dish	-99.0
3360	cheat ni	-99.0
3339	clean nroom	-99.0
3338	superior delux valley face room	-99.0
3337	point room servic	-99.0
3336	travel desk	-99.0
3334	sure everyon	-99.0
3355	tv recept poor bathroom water clog n1st day	-99.0
3333	morn person attent	-99.0
3361	shirt r	-99.0
3332	lawn nfresh food nprompt servic	-99.0
3331	huge properti nclean pool	-99.0
3330	good featur	-99.0
3329	panchgani cool breez valley view open lawn minut	-99.0
3359	prize r	-99.0
3340	gener backup	-99.0
3341	ac ngood food	-99.0
3348	peopl cleartrip	-99.0
3354	corridor fridg	-99.0
3353	secur nbathroom window	-99.0
3352	mainten ni	-99.0
3351	pathet food	-99.0
3350	nsriniva popuri	-99.0
3349	trust ni	-99.0
3347	elit custom	-99.0

	word	pmi
3342	sumptuou spread	-99.0
3357	photo graph	-99.0
3346	innoc travel	-99.0
3358	amaz thing	-99.0
3328	friendli guy	-99.0
3344	horribl hotel unnecessarili	-99.0
3343	buffet breakfast nswim pool	-99.0
3345	hotel peopl	-99.0
0	hot water	-99.0
3327	parti game	-99.0
3298	pleasant memori thing	-99.0
3305	nyt stay	-99.0
3304	clean nfood decent nbreakfast	-99.0

In [72]:
```
pmiposlist_bot, pmineglist_bot, pmidf_bot = pmiForAllCal(finaldf.loc[finaldf['hotelName']==hote
lRatingDfGt.tail(1)['hotelName'].values[0]],'groundTruth',topk)
pmiposlist_bot.sort_values(by='pmi', ascending=0).head(50)
```

```
100%|██████████| 1500/1500 [00:19<00:00, 77.52it/s]
```

Out[72]:

	word	pmi
0	hotel	-99
997	helping	-99
1006	inr	-99
1005	drinks	-99
1004	carry	-99
1003	evenings	-99
1002	present	-99
1001	behind	-99
1000	august	-99
999	closed	-99
998	atleast	-99
996	whether	-99
985	blankets	-99
995	nwould	-99
994	plenty	-99
993	corner	-99
992	hand	-99
991	true	-99
990	veranda	-99
989	school	-99
988	wooden	-99
987	baby	-99
1007	pls	-99
1008	arrange	-99
1009	july	-99
1010	chef	-99
1029	decision	-99
1028	requested	-99
1027	reaching	-99
1026	satisfied	-99
1025	problems	-99
1024	advantage	-99
1023	move	-99
1022	december	-99
1021	bunglow	-99
1020	electricity	-99
1019	dec	-99

	word	pmi
1018	krishna	-99
1017	wall	-99
1016	soft	-99
1015	valleys	-99
1014	secluded	-99
1013	terra	-99
1012	birthday	-99
1011	shop	-99
986	masala	-99
984	husband	-99
1031	stairs	-99
950	saying	-99
959	security	-99

```
In [73]: pmineglist_bot.sort_values(by='pmi', ascending=0).head(50)
```

Out[73]:

	word	pmi
167	going	0.0
1021	bunglow	0.0
26	location	0.0
1019	dec	0.0
228	without	0.0
1328	taker	0.0
1329	bill	0.0
34	get	0.0
36	water	0.0
793	attitude	0.0
38	even	0.0
680	rude	0.0
234	help	0.0
552	actually	0.0
136	care	0.0
53	night	0.0
246	kept	0.0
145	went	0.0
1174	2018	0.0
522	lawn	0.0
330	anything	0.0
65	need	0.0
1076	badminton	0.0
1020	electricity	0.0
35	really	0.0
23	would	0.0
3	room	0.0
206	walk	0.0
488	issues	0.0
11	one	0.0
5	place	0.0
1118	asking	0.0
950	saying	0.0
200	bathrooms	0.0
385	almost	0.0
199	average	0.0
107	outside	0.0

	word	pmi
653	caretaker	0.0
763	stop	0.0
565	let	0.0
999	closed	-99.0
1001	behind	-99.0
1002	present	-99.0
1026	satisfied	-99.0
998	atleast	-99.0
1028	requested	-99.0
997	helping	-99.0
1003	evenings	-99.0
1000	august	-99.0
1027	reaching	-99.0

Answer 4(c):

- To handle the case of px/py =0, the pmi is set of -99 to be able to discern the terms relevant to the top/bottom hotel reviews from the other terms
- From the phrase analysis, the phrases such as the 'reason price', 'amaz hospit', 'delici food', 'perfect locat' indicate the top most hotel- has resonable price, good food and hospitable staff
- Since all the reviews of the best hotel were positive, no relevant negative pharases/ terms were obtained from the top hotel
- Since, the bottomost hotel has phrase such as 'hotel staff' and other terms such as the 'water', 'electricity' and bill, it indicated that the hotel staff were not good. Moreover, people had bad experince with water and electricity in this hotel

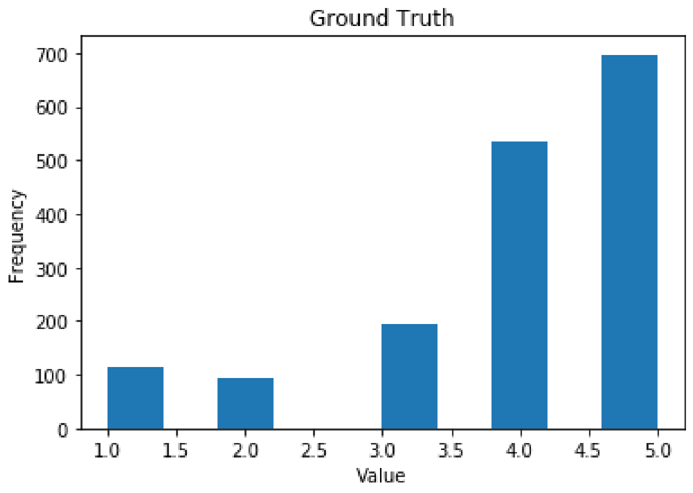
Q5 General Plots

Q5 (a) Histogram

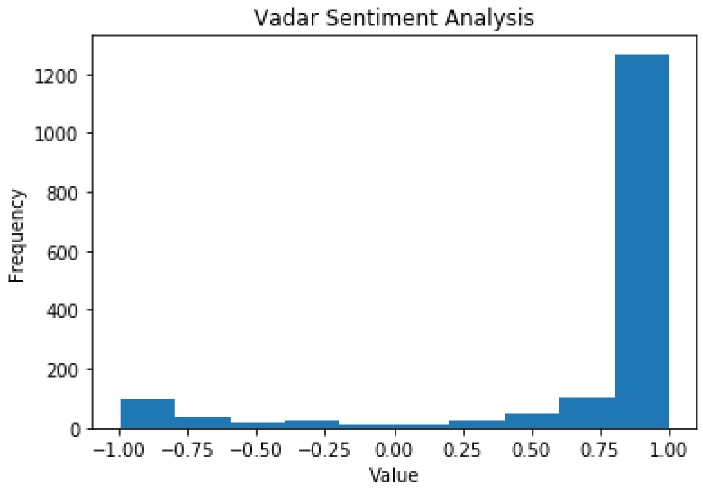
(a)

```
In [0]: def getHistogram(measure, title):
            if measure=='both':
                x = [finaldf['ratingScore'].values/5]
                y = [finaldf['vader'].values]
                bins = np.linspace(-1, 1, 100)
                plt.title(title)
                plt.hist(x, bins, label='ground truth')
                plt.hist(y, bins, label='vader')
                plt.legend(loc='upper right')
                plt.xlabel("Value")
                plt.ylabel("Frequency")
                plt.show()

            else:
                plt.hist(finaldf[measure].values)
                plt.title(title)
                plt.xlabel("Value")
                plt.ylabel("Frequency")
                fig = plt.gcf()
```

```
In [75]: getHistogram('ratingScore', 'Ground Truth')
```



```
In [76]: getHistogram('vader', 'Vadar Sentiment Analysis')
```

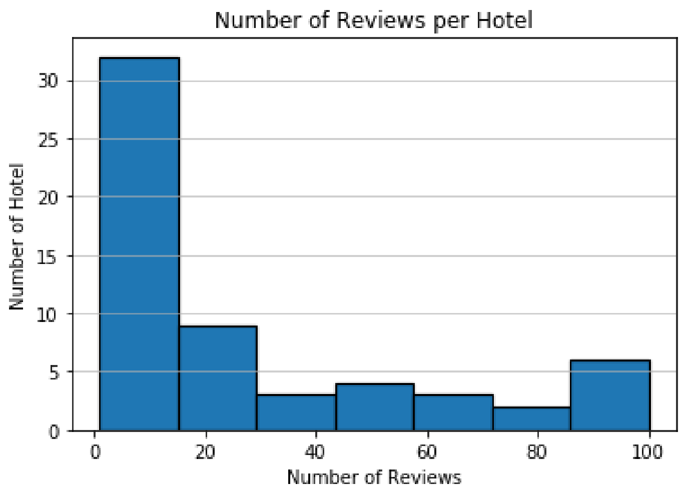


Answer 5(a):

- The distribution of the ratings were similar for the ground truth and the vader compound scores. The ground truth rating (4/5) had a vader score of 1
- It was an unbalanced dataset with the majority of the ratings being positive. It was expected as the top 50 PMIs for the negative phrases included positive phrases with the negative PMIs

(b)

```
In [77]: ratings=getNoOfRatings(hotelDf)
```



```
In [78]: ratings.describe()
```

Out[78]:

	no_of_reviews
count	59.000000
mean	27.694915
std	31.985586
min	1.000000
25%	5.000000
50%	12.000000
75%	44.500000
max	100.000000

Answer 5(b):

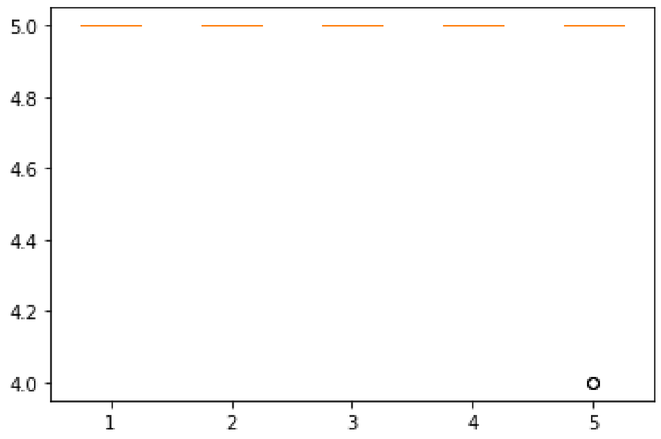
- The maximum number of reviews were 100, this was expected as the maximum number of reviews per hotel was restricted to 100
- The data had in all reviews of 59 hotels with the majority of the hotels having less than 20 reviews

Q5 (b) Boxplots

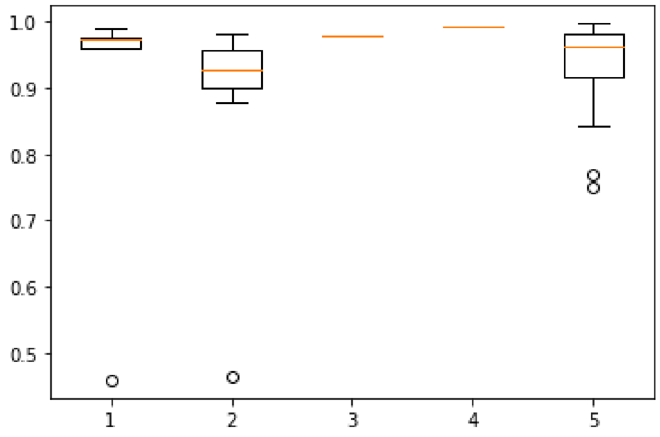
(a)

```
In [79]: #Plot top 5 side-by-side boxplot for top 5 ground truth rated hotel
         tp5gthotel = hotelRatingDfGt.sort_values('avgRatingScore',ascending=0).head(5)
         tp5gthotel['hotelName'].values

Out[79]: array(['Basilica Rediscover Serenity', 'The Loft', 'Tranquil Treasure',
                'Jivanta Hotel', 'Magnus Caverns Resort'], dtype=object)
```

```
In [80]: Basilica  = finaldf.loc[finaldf['hotelName'] == tp5gthotel['hotelName'].values[0]]['ratingScor
         e']
         Loft = finaldf.loc[finaldf['hotelName'] == tp5gthotel['hotelName'].values[1]]['ratingScore']
         Tranquil  = finaldf.loc[finaldf['hotelName'] == tp5gthotel['hotelName'].values[2]]['ratingScor
         e']
         Jivanta  = finaldf.loc[finaldf['hotelName'] == tp5gthotel['hotelName'].values[3]]['ratingScore'
         ]
         Magnus  = finaldf.loc[finaldf['hotelName'] == tp5gthotel['hotelName'].values[4]]['ratingScore']

         data = [Basilica, Loft, Tranquil, Jivanta, Magnus]
         # multiple box plots on one figure
         plt.figure()
         plt.boxplot(data)
         plt.show()
```

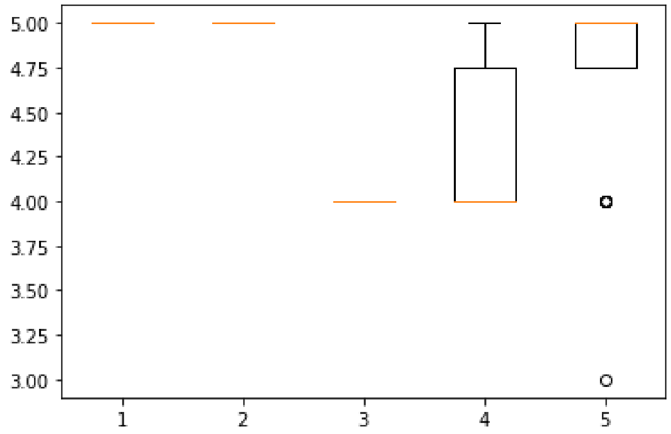


```
In [81]: Basilica  = finaldf.loc[finaldf['hotelName'] == tp5gthotel['hotelName'].values[0]]['vader']
         Loft = finaldf.loc[finaldf['hotelName'] == tp5gthotel['hotelName'].values[1]]['vader']
         Tranquil  = finaldf.loc[finaldf['hotelName'] == tp5gthotel['hotelName'].values[2]]['vader']
         Jivanta  = finaldf.loc[finaldf['hotelName'] == tp5gthotel['hotelName'].values[3]]['vader']
         Magnus  = finaldf.loc[finaldf['hotelName'] == tp5gthotel['hotelName'].values[4]]['vader']

         data = [Basilica, Loft, Tranquil, Jivanta, Magnus]
         # multiple box plots on one figure
         plt.figure()
         plt.boxplot(data)
         plt.show()
```

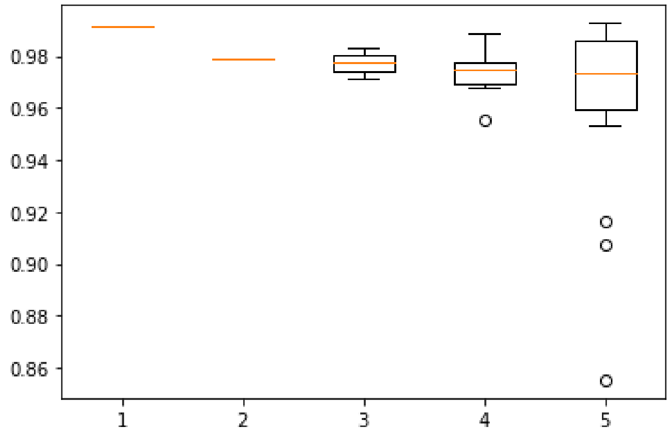


```
In [82]: #Plot top 5 side-by-side boxplot for top 5 Vader rated hotel
         tp5vdhotel = hotelRatingDfVd.sort_values('avgRatingScore',ascending=0).head(5)
         tp5vdhotel['hotelName'].values

Out[82]: array(['Jivanta Hotel', 'Tranquil Treasure',
                'JenJon Holiday Homes Panchgani', 'Mayur Agro Park',
                'SaffronStays Verandah by the Valley'], dtype=object)
```

```
In [83]: Jivanta  = finaldf.loc[finaldf['hotelName'] == tp5vdhotel['hotelName'].values[0]]['ratingScore'
         ]
         Tranquil  = finaldf.loc[finaldf['hotelName'] == tp5vdhotel['hotelName'].values[1]]['ratingScor
         e']
         JenJon   = finaldf.loc[finaldf['hotelName'] == tp5vdhotel['hotelName'].values[2]]['ratingScore'
         ]
         Mayur   = finaldf.loc[finaldf['hotelName'] == tp5vdhotel['hotelName'].values[3]]['ratingScore']
         SaffronStays   = finaldf.loc[finaldf['hotelName'] == tp5vdhotel['hotelName'].values[4]]['rating
         Score']

         data = [Jivanta, Tranquil, JenJon, Mayur, SaffronStays]
         # multiple box plots on one figure
         plt.figure()
         plt.boxplot(data)
         plt.show()
```



```
In [84]: Jivanta  = finaldf.loc[finaldf['hotelName'] == tp5vdhotel['hotelName'].values[0]]['vader']
         Tranquil  = finaldf.loc[finaldf['hotelName'] == tp5vdhotel['hotelName'].values[1]]['vader']
         JenJon   = finaldf.loc[finaldf['hotelName'] == tp5vdhotel['hotelName'].values[2]]['vader']
         Mayur   = finaldf.loc[finaldf['hotelName'] == tp5vdhotel['hotelName'].values[3]]['vader']
         SaffronStays   = finaldf.loc[finaldf['hotelName'] == tp5vdhotel['hotelName'].values[4]]['vader'
         ]

         data = [Jivanta, Tranquil, JenJon, Mayur, SaffronStays]
         # multiple box plots on one figure
         plt.figure()
         plt.boxplot(data)
         plt.show()
```



(b)

```
In [0]: def getMeanVar(df, tp5gthotel, measure):
          tp5df=df.loc[df['hotelName'].isin(tp5gthotel['hotelName'].values)]#['vader']
          meandf=tp5df.groupby('hotelName')[measure].mean().to_frame(name='mean').reset_index()
          vardf=tp5df.groupby('hotelName')[measure].var().to_frame(name='variance').reset_index().filln
        a(0)
          meanvardf=meandf.merge(vardf, on='hotelName')
          return meanvardf
```

```
In [86]: getMeanVar(finaldf, tp5gthotel, 'ratingScore')
```

Out[86]:

	hotelName	mean	variance
0	Basilica Rediscover Serenity	5.000000	0.000000
1	Jivanta Hotel	5.000000	0.000000
2	Magnus Caverns Resort	4.833333	0.151515
3	The Loft	5.000000	0.000000
4	Tranquil Treasure	5.000000	0.000000

```
In [87]: getMeanVar(finaldf, tp5gthotel, 'vader')
```

Out[87]:

	hotelName	mean	variance
0	Basilica Rediscover Serenity	0.871580	0.053351
1	Jivanta Hotel	0.991800	0.000000
2	Magnus Caverns Resort	0.926467	0.007642
3	The Loft	0.869586	0.032895
4	Tranquil Treasure	0.979100	0.000000

Answer 5(b):

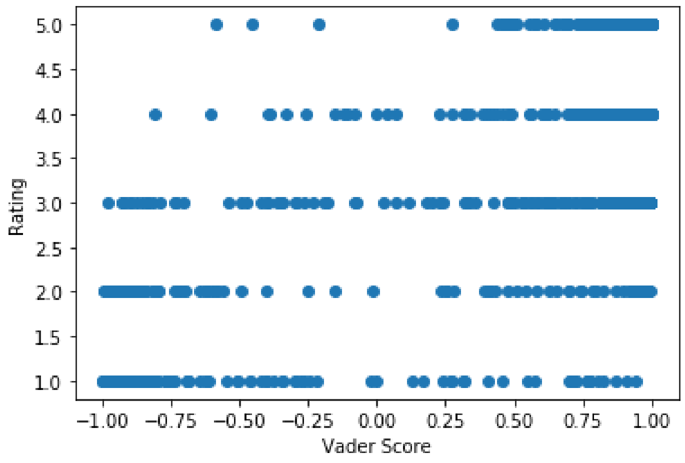
- Boxplot is more informative compared to that of the mean and variance. It gives the information of the median, quartiles, spread and the range (minimum and maximum). The only data only covered in the boxplot is the mean, however, since the distribution is skewed, median is more information compared to the mean

(c)

Q5 (c) Scatterplots and heatmaps

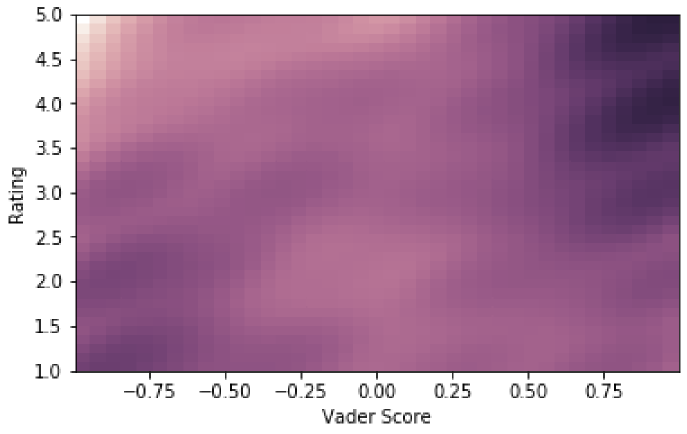
(a)

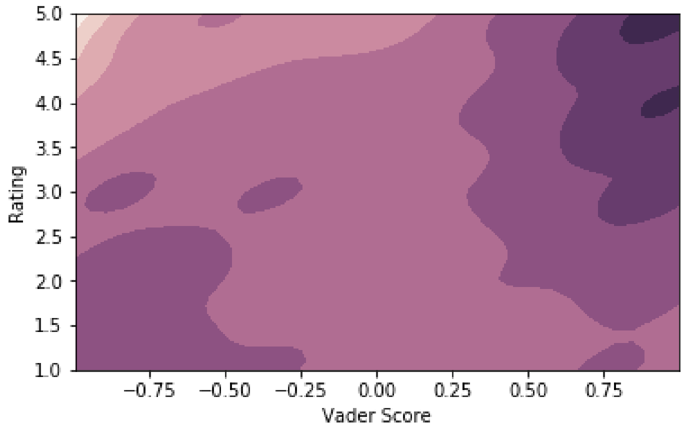
```
In [88]: y = finaldf['ratingScore'].values
         x = finaldf['vader'].values
         plt.plot(x, y,"o")
         plt.ylabel('Rating')
         plt.xlabel('Vader Score')

Out[88]: Text(0.5, 0, 'Vader Score')
```



In [89]:
```
k = gaussian_kde(np.vstack([x, y]))
xi, yi = np.mgrid[x.min():x.max():x.size**0.5*1j,y.min():y.max():y.size**0.5*1j]
zi = k(np.vstack([xi.flatten(), yi.flatten()]))

cmap = sns.cubehelix_palette(light=1, as_cmap=True)
fig = plt.figure(figsize=(6,8))
ax1 = fig.add_subplot(211)
ax2 = fig.add_subplot(212)

ax1.pcolormesh(xi, yi, np.log10(zi.reshape(xi.shape)), cmap=cmap)
ax2.contourf(xi, yi, np.log10(zi.reshape(xi.shape)), cmap=cmap)

ax1.set_xlim(x.min(), x.max())
ax1.set_ylim(y.min(), y.max())
ax2.set_xlim(x.min(), x.max())
ax2.set_ylim(y.min(), y.max())

ax1.set_xlabel('Vader Score')
ax1.set_ylabel('Rating')

ax2.set_xlabel('Vader Score')
ax2.set_ylabel('Rating')
```

Out[89]:
```
Text(0, 0.5, 'Rating')
```



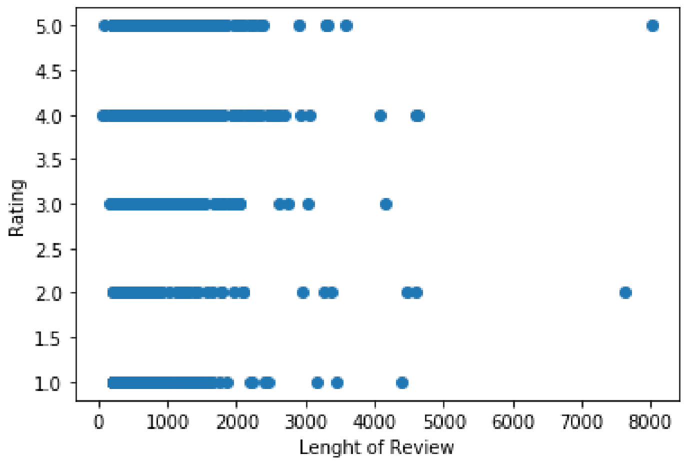


Answer 5(c)(a):

- Scatterplot gives the relation between the Vader score and the Star Ratings. For example, for the reviews with Star Rating of 5 majority of the vader score was 1. Similarily, majority of the reviews with the Startratings 1 had vader score of -1. The distribution of the vader score was more scattered for the reviews with star rating 3
- Heatmap shows the strength of relationship between the vader score and star ratings. Stronger the relation i.e. denser the points for a given rating and vader score, the darker the heatmap gets around that region.

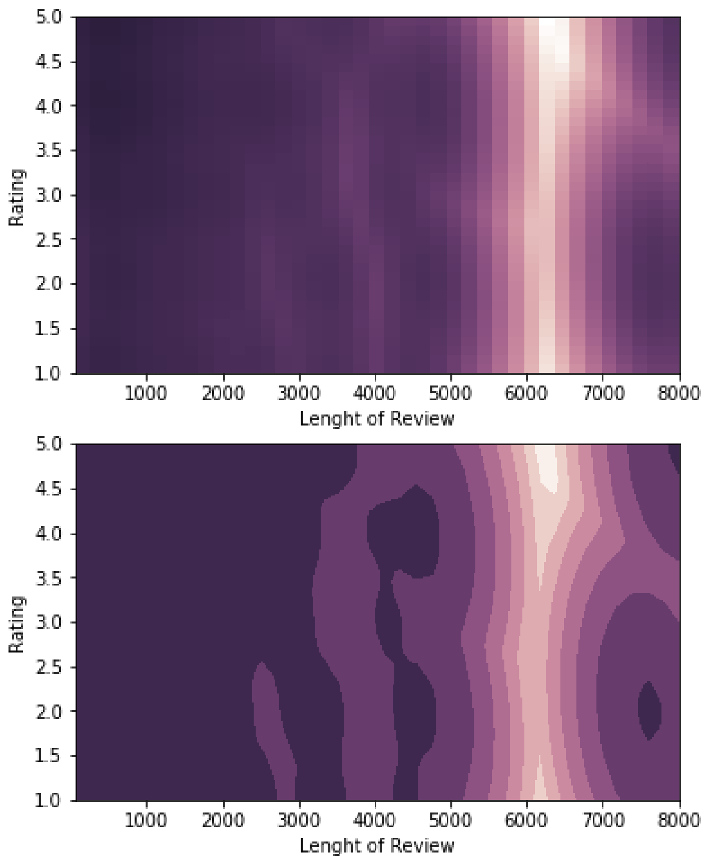
(b)

```
In [90]: y = finaldf['ratingScore'].values
         x= finaldf['reviewCol'].map(len).values
         plt.plot(x, y,"o")
         plt.ylabel('Rating')
         plt.xlabel('Lenght of Review')

Out[90]: Text(0.5, 0, 'Lenght of Review')
```

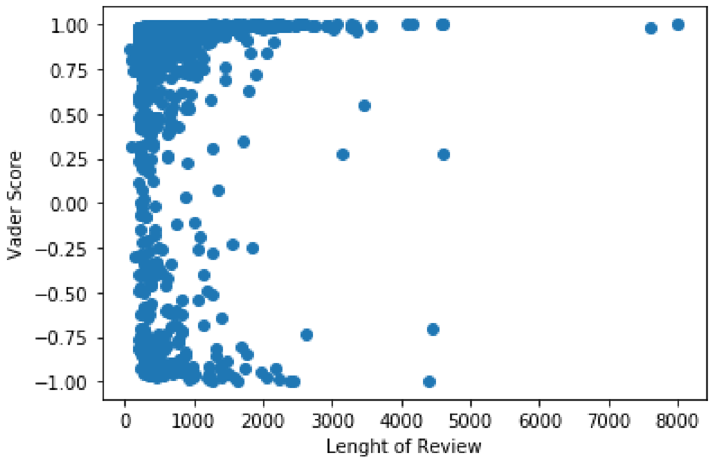


In [91]:
```
k = gaussian_kde(np.vstack([x, y]))
xi, yi = np.mgrid[x.min():x.max():x.size**0.5*1j,y.min():y.max():y.size**0.5*1j]
zi = k(np.vstack([xi.flatten(), yi.flatten()]))

cmap = sns.cubehelix_palette(light=1, as_cmap=True)
fig = plt.figure(figsize=(6,8))
ax1 = fig.add_subplot(211)
ax2 = fig.add_subplot(212)

ax1.pcolormesh(xi, yi, np.log10(zi.reshape(xi.shape)), cmap=cmap)
ax2.contourf(xi, yi, np.log10(zi.reshape(xi.shape)), cmap=cmap)

ax1.set_xlim(x.min(), x.max())
ax1.set_ylim(y.min(), y.max())
ax2.set_xlim(x.min(), x.max())
ax2.set_ylim(y.min(), y.max())

ax1.set_xlabel('Lenght of Review')
ax1.set_ylabel('Rating')

ax2.set_xlabel('Lenght of Review')
ax2.set_ylabel('Rating')
```

Out[91]:
```
Text(0, 0.5, 'Rating')
```

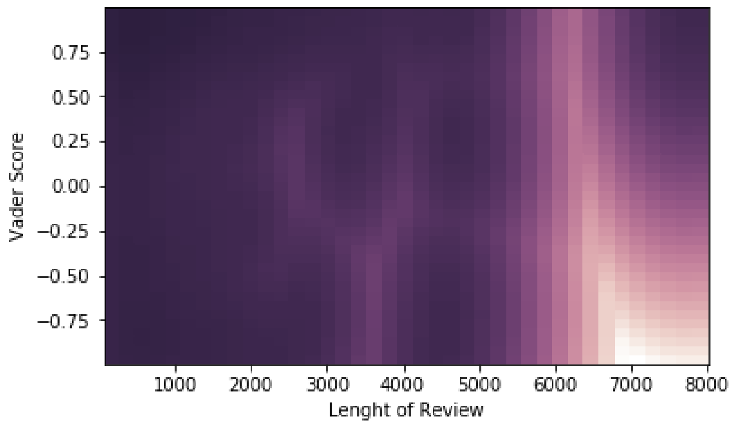


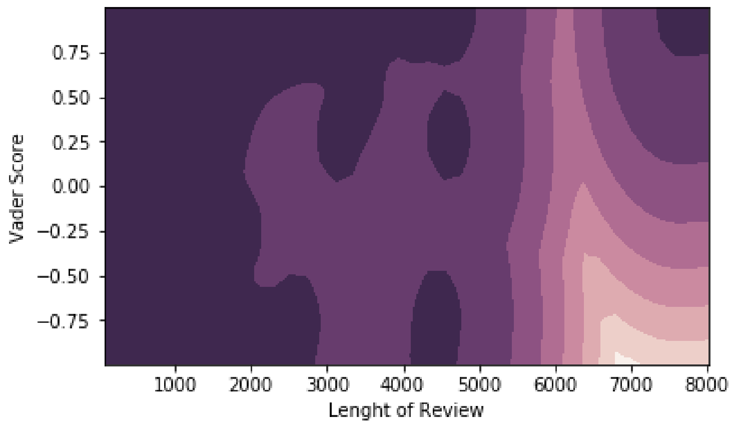
```
In [92]: y = finaldf['vader'].values
         x= finaldf['reviewCol'].map(len).values
         plt.plot(x, y,"o")
         plt.ylabel('Vader Score')
         plt.xlabel('Lenght of Review')

Out[92]: Text(0.5, 0, 'Lenght of Review')
```



In [93]:
```
k = gaussian_kde(np.vstack([x, y]))
xi, yi = np.mgrid[x.min():x.max():x.size**0.5*1j,y.min():y.max():y.size**0.5*1j]
zi = k(np.vstack([xi.flatten(), yi.flatten()]))

cmap = sns.cubehelix_palette(light=1, as_cmap=True)
fig = plt.figure(figsize=(6,8))
ax1 = fig.add_subplot(211)
ax2 = fig.add_subplot(212)

ax1.pcolormesh(xi, yi, np.log10(zi.reshape(xi.shape)), cmap=cmap)
ax2.contourf(xi, yi, np.log10(zi.reshape(xi.shape)), cmap=cmap)

ax1.set_xlim(x.min(), x.max())
ax1.set_ylim(y.min(), y.max())
ax2.set_xlim(x.min(), x.max())
ax2.set_ylim(y.min(), y.max())

ax1.set_xlabel('Lenght of Review')
ax1.set_ylabel('Vader Score')

ax2.set_xlabel('Lenght of Review')
ax2.set_ylabel('Vader Score')
```

Out[93]:
```
Text(0, 0.5, 'Vader Score')
```





Answer 5(c)(b):

- There is no trend in the length of reviews and the correspoding score/rating. Majority of the reviews had a length of less than 2000. The length of the reviews with vader score of 1 was comparatively higher thans that of the reviews with the vader score of -1

(c)

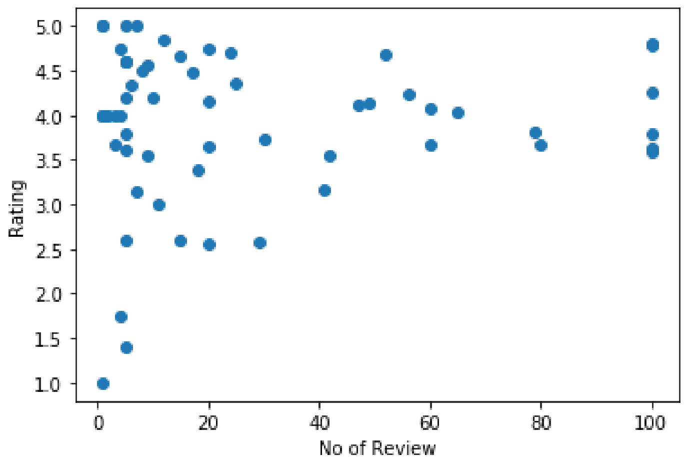
```
In [0]: gtrat=ratings.merge(hotelRatingDfGt, on='hotelName')
        reviewsRatings=gtrat.merge(hotelRatingDfVd, on='hotelName')
        reviewsRatings.rename(columns={'avgRatingScore_x': 'avgRatingScore', 'avgRatingScore_y': 'avgVa
        der', 'no_of_reviews': 'noOfReviews'}, inplace=True)
```

```
In [95]: reviewsRatings.head()
```

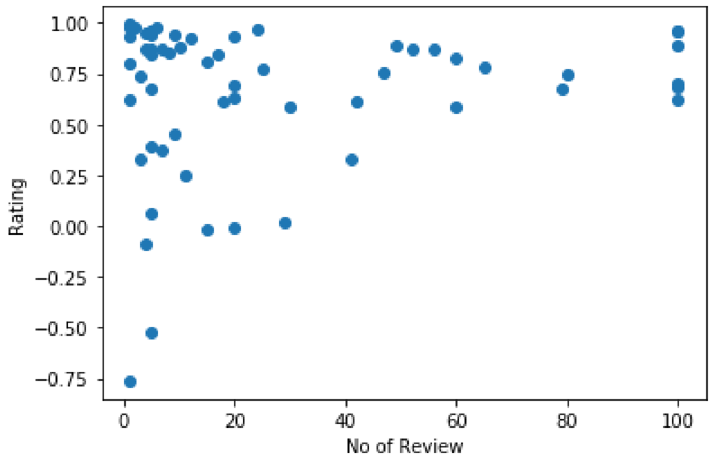
Out[95]:

	hotelName	noOfReviews	avgRatingScore	avgVader
0	Alliance Tents and Accommodations	4	1.75	-0.088625
1	Animish Bungalow	1	4.00	0.935300
2	Ashirwad Bungalow	1	1.00	-0.761600
3	Basilica Rediscover Serenity	5	5.00	0.871580
4	Bellevue Resort	25	4.36	0.771240

```
In [96]: y = reviewsRatings['avgRatingScore'].values
         x= reviewsRatings['noOfReviews'].values
         plt.plot(x, y,"o")
         plt.ylabel('Rating')
         plt.xlabel('No of Review')

Out[96]: Text(0.5, 0, 'No of Review')
```



```
In [97]: y = reviewsRatings['avgVader'].values
         x= reviewsRatings['noOfReviews'].values
         plt.plot(x, y,"o")
         plt.ylabel('Rating')
         plt.xlabel('No of Review')

Out[97]: Text(0.5, 0, 'No of Review')
```



Answer 5(c)(c):

- There is no trend in the number of reviews and the correspoding score/rating. There were five hotels with 100 reviews signifying their popularity. Also, all these hotels had positive reviews.