Sentiment distribution and normalized sentiment frequencies

June 21, 2016

In [1]:
```
# The usual preamble
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
import pylab
from __future__ import division


# Make the graphs a bit prettier, and bigger
pd.set_option('display.mpl_style', 'default')
plt.rcParams['figure.figsize'] = (15, 5)


# This is necessary to show lots of columns in pandas 0.12.
# Not necessary in pandas 0.13.
pd.set_option('display.width', 5000)
pd.set_option('display.max_columns', 60)
```

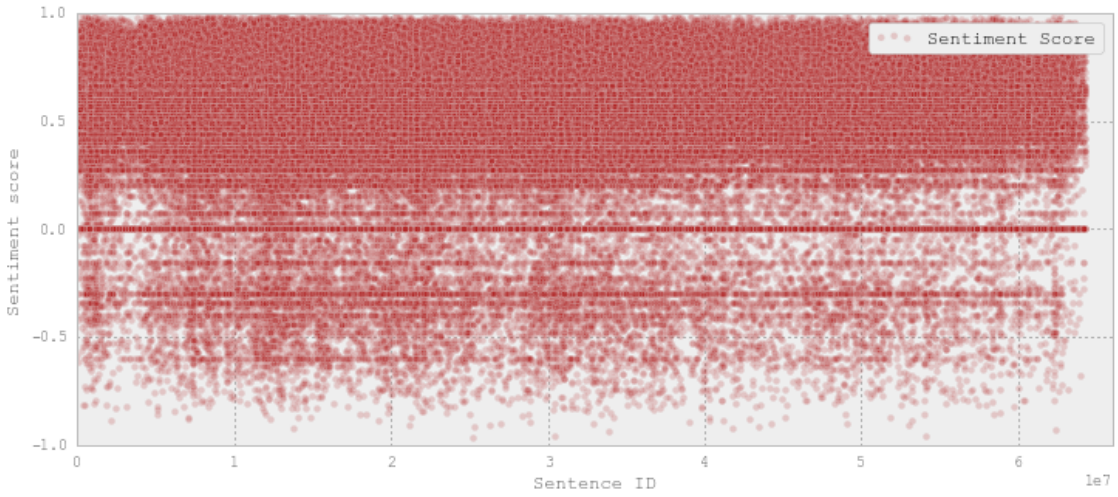
```
c:\python27\lib\site-packages\IPython\core\interactiveshell.py:2885: FutureWarning:
mpl_style had been deprecated and will be removed in a future version.
Use `matplotlib.pyplot.style.use` instead.

  exec(code_obj, self.user_global_ns, self.user_ns)
```

1 Distribution of sentiment scores of each sentence

First we read the file where all the sentiment scores are saved and then plot their distribution based on the sentence and its corresponding sentiment score. Three main insights can be drawn from this graph: * We can clearly see that most of the scores are located in the positive zone, above zero. * Many sentences have 0 sentiment and this case is treated in the results chapter. * There is a trend to avoid areas with slight

In [2]:
```
file = pd.read_csv('C:/Python27/output_improved_AMS.csv')
d=file[['Review ID','Sentiment score']].plot(kind='scatter',
```

```
    x='Review ID', y='Sentiment score', color='FireBrick',
    label='Sentiment Score', alpha=0.2, figsize=(12,5))
plt.axis([0,65999999,-1.0,1.0])
d.set_xlabel("Sentence ID")
plt.show()
```



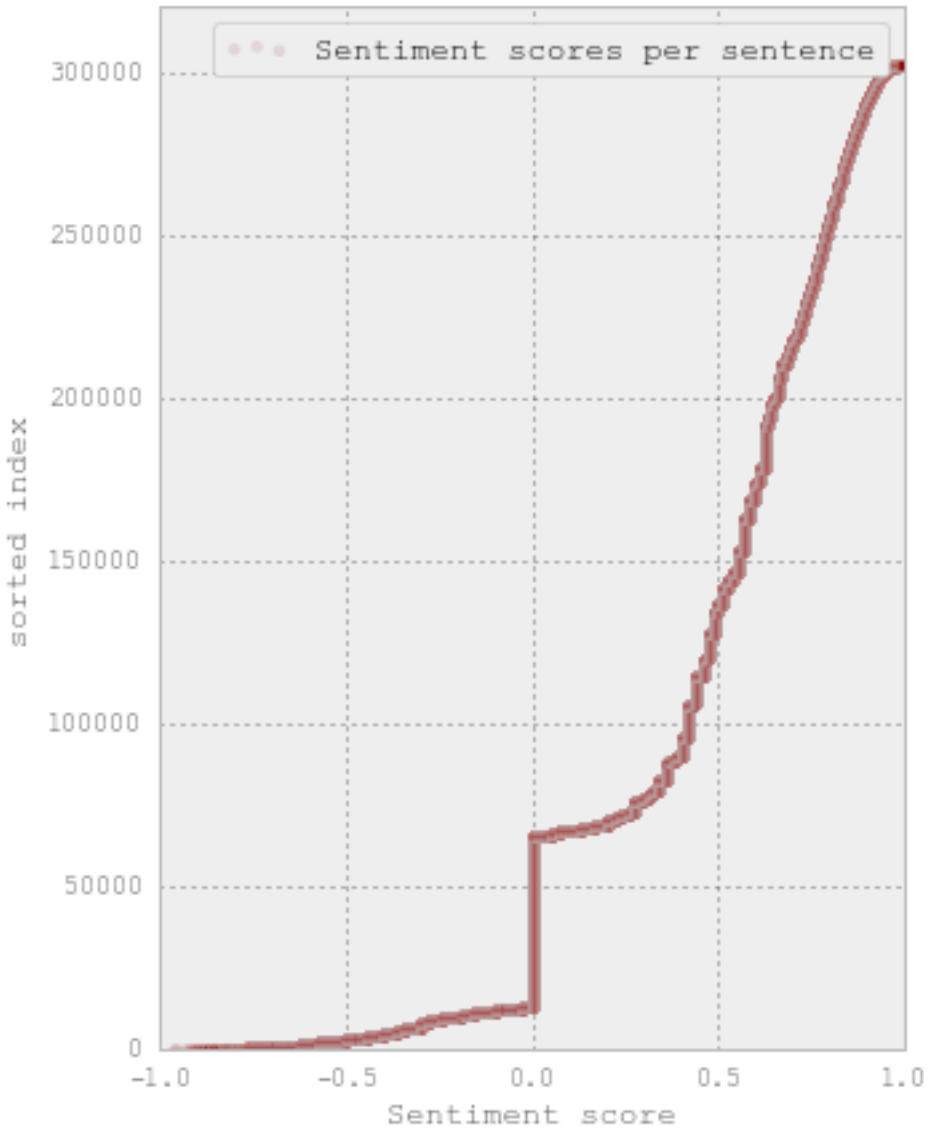
2 Sorted array of sentiment scores

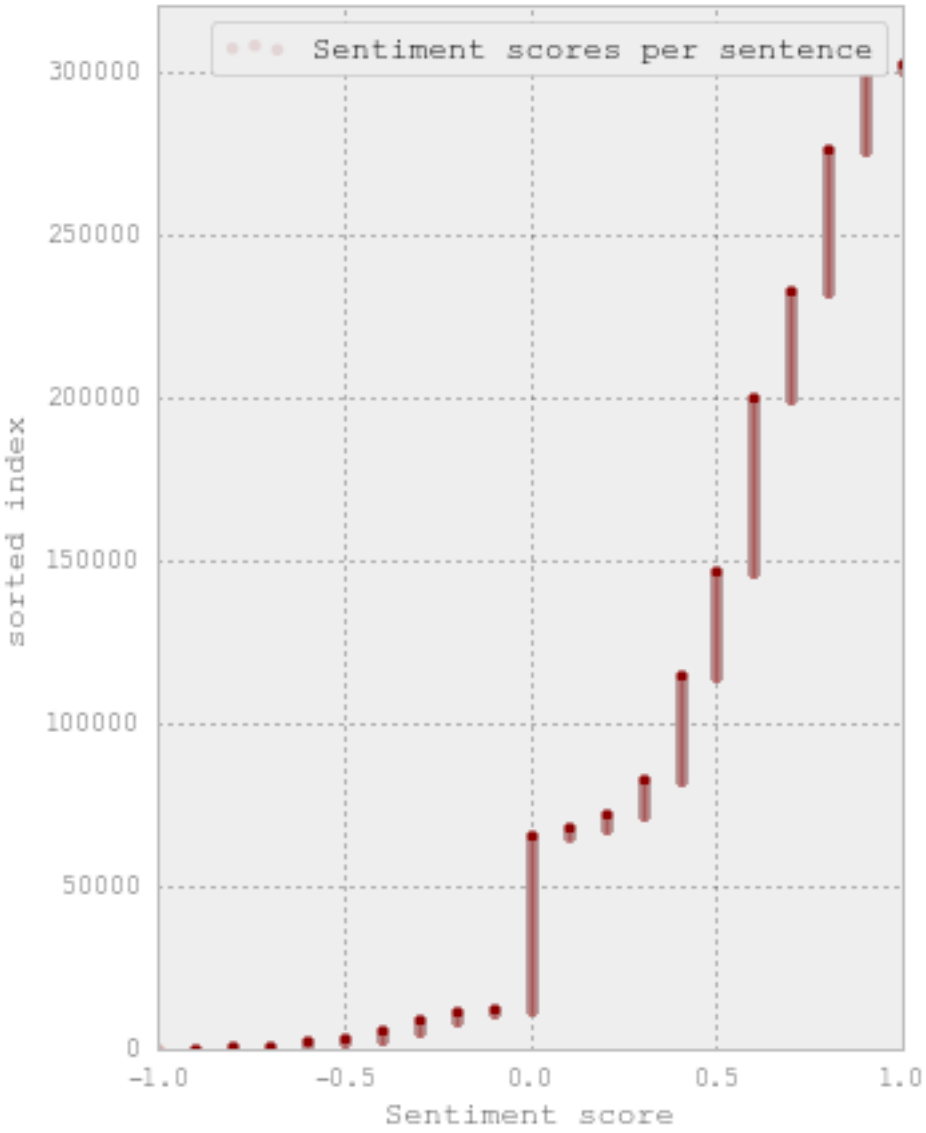
In the first graph the sorted sentiment scores are shown. In this way the slope of the curve indicates that there are much more positive scores than negative ones. Also the amount of sentences with sentiment 0 is more obvious. So this is just another form for plotting the conclusions drawn above.

The second graph shows the same sorted sentiment values but this time they are rounded with 1 digit. This plot shows not only the frequency of scores but also the trend of the curve.

```
In [19]: # Sentiment scores in another form of visualizing

         ak=file[['Sentiment score']].sort_values('Sentiment score',
                                        axis=0, ascending=True)
         ak.reset_index(level=0, inplace=True)
         ak['index']=ak.index
         i=ak.plot(kind='scatter', y='index', x='Sentiment score',
                       color='DarkRed', label='Sentiment scores per sentence',
                   alpha=0.1, figsize=(5,7))
         plt.axis([-1, 1, 0, 320000])
         i.set_ylabel("sorted index")
         plt.show()

         # Sentiment scores in another form of visualizing
         ak=file[['Sentiment score']].sort_values('Sentiment score',
                 axis=0, ascending=True).round(1)
```

```
ak.reset_index(level=0, inplace=True)
ak['index']=ak.index
ii=ak.plot(kind='scatter', y='index', alpha=0.1,
           x='Sentiment score', color='DarkRed',
          label='Sentiment scores per sentence', figsize=(5,7))
ii.set_ylabel("sorted index")
plt.axis([-1, 1, 0, 320000])
plt.show()
```





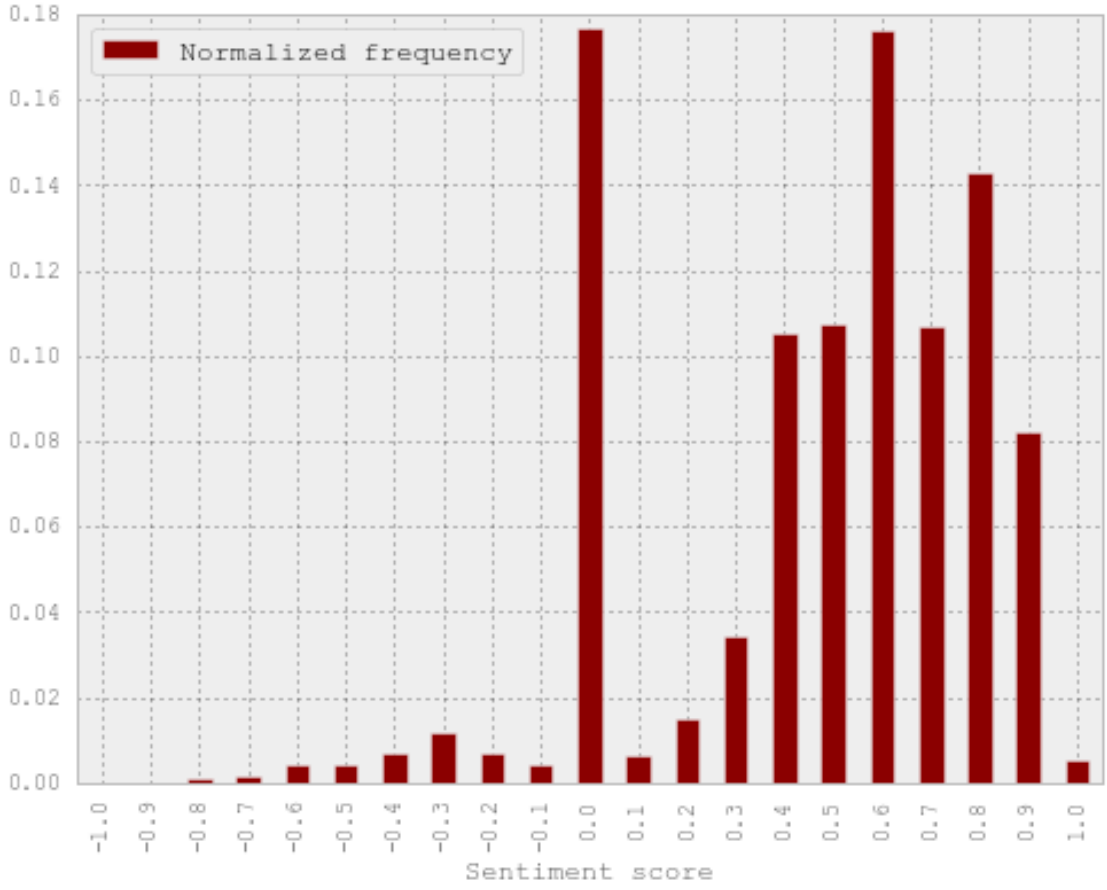
3 Normalized frequencies of sentiment scores rounded by 1

The graph shows the normalized frequencies of each sentimnet score rounded by 1 digit. We see that the most frequent one are 0, then 0.6 and 0.8. Besides the presence of zero explained in results, the presence of two relatively high sentimnet scores shows the trend of having high ratings. In the literature this is called the J-shape.

```
In [4]: # Normalized frequency of sentiment scores
        aa=ak[['Sentiment score']].round(1)
        ab=aa['Sentiment score'].value_counts(normalize=True, sort=False)
        af=pd.DataFrame({'Sentiment score':ab.index,
                         'Normalized frequency':ab.values})
```

```
h=af.sort_values('Sentiment score', axis=0, ascending=True)
h.plot(kind='bar',x='Sentiment score', y='Normalized frequency',
       figsize=(8,6), color='DarkRed')
```

```
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x9d65630>
```

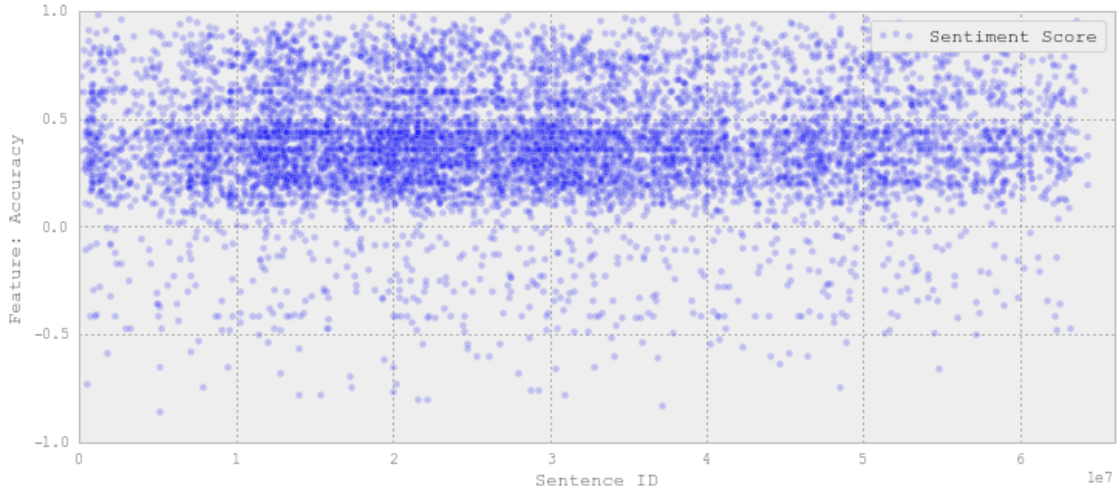


4 Distribution of sentiment scores per each feature

The distribution pattern of the sentimnet scores as above is repeated for all the features particularly. The aim is to find out how these patterns differ from each other. The following observations are noticed: * Feature ** Location** has the most similar distribution to the sentiment scores per sentence seen above. This pattern can be explained with the fact that **Location** is the most mentioned feature as we will see later on. * **Cleanliness** and **Value** have similar distributions, whose sentiment scores are located in two main zones in the positive area by distriguishing in slightly positive and strong positive. * We see that for **Accuracy, Communication, Cleanliness** and **Value** the presence of sentences with no sentiment is not very notable.

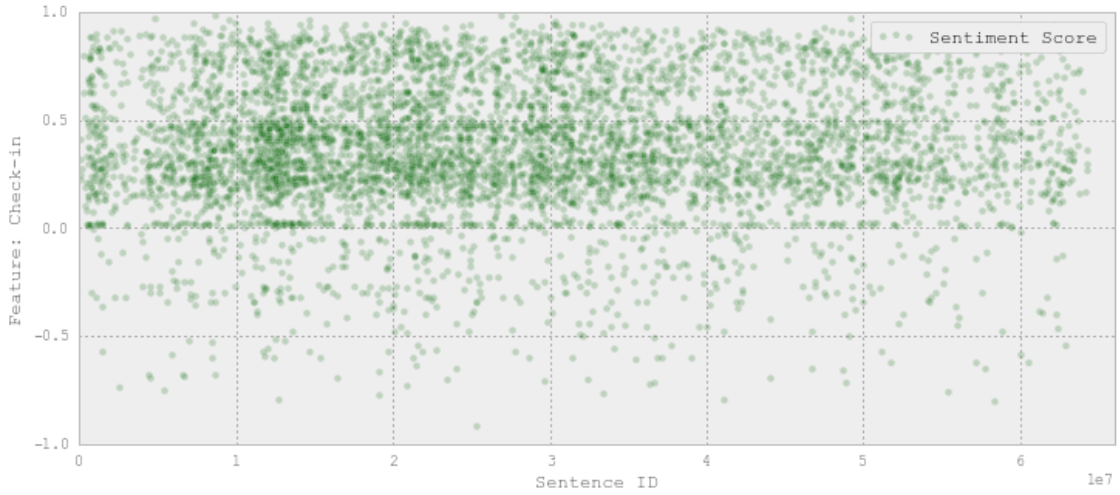
4.1 Accuracy

```
In [5]: acc=file[file['Feature: Accuracy']!=0]
        acc[['Review ID','Feature: Accuracy']].plot(kind='scatter',
            x='Review ID', y='Feature: Accuracy', color='Blue',
            label='Sentiment Score', alpha=0.2,figsize=(12,5)).set_xlabel("Sentence
        plt.axis([0,65999999,-1.0,1.0])
        plt.show()
        acr=acc.round(1)
```



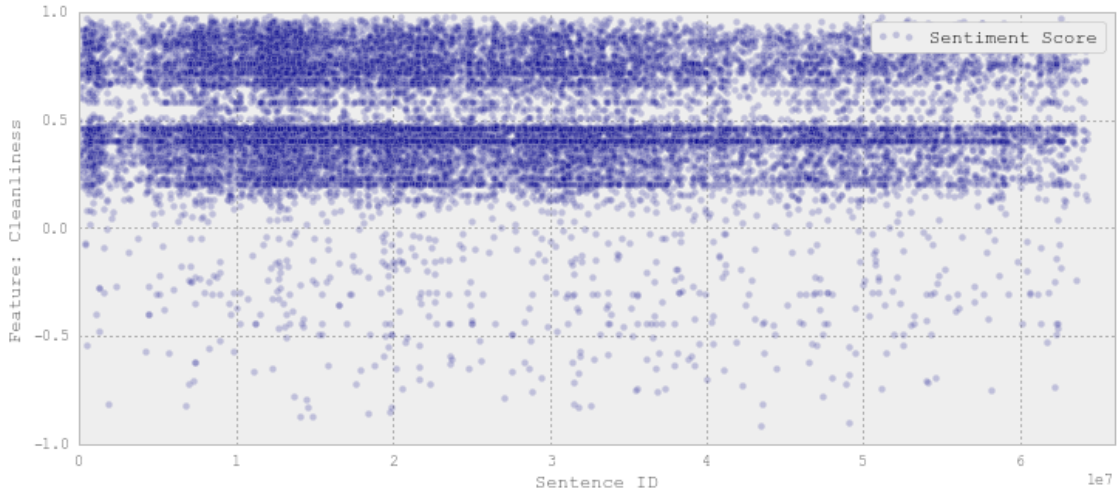
4.2 Check-in

```
In [6]: chk=file[file['Feature: Check-in']!=0]
        chk[['Review ID','Feature: Check-in']].plot(kind='scatter',
              x='Review ID', y='Feature: Check-in',color='DarkGreen',
              label='Sentiment Score', alpha=0.2, figsize=(12,5)).set_xlabel("Sente
        plt.axis([0,65999999,-1.0,1.0])
        plt.show()
        chkr=chk.round(1)
```



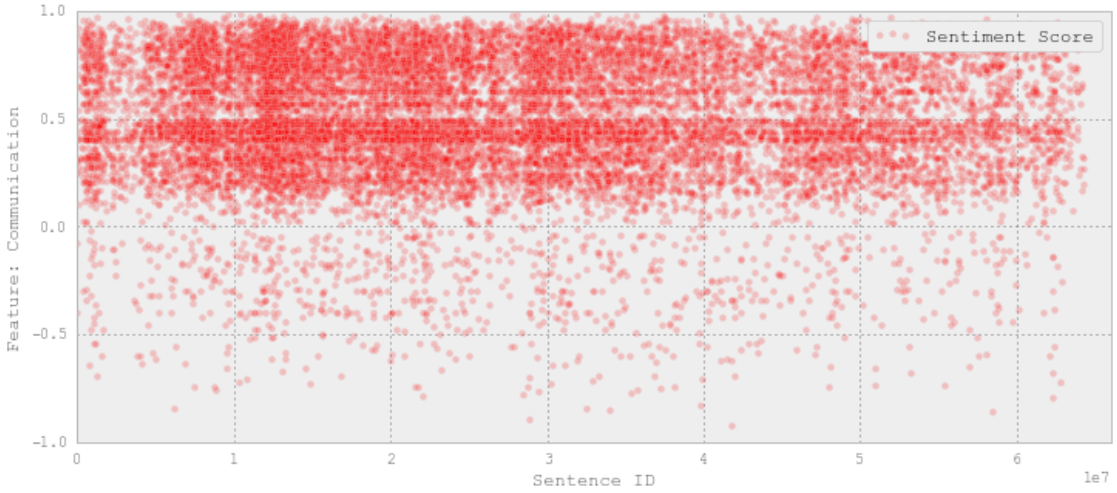
4.3 Cleanliness

```
In [7]: cle=file[file['Feature: Cleanliness']!=0]
        cle[['Review ID','Feature: Cleanliness']].plot(kind='scatter',
              x='Review ID', y='Feature: Cleanliness', color='DarkBlue',
               label='Sentiment Score', alpha=0.2, figsize=(12,5)).set_xlabel("Sent
        plt.axis([0,65999999,-1.0,1.0])
        plt.show()
        cler=cle.round(1)
```



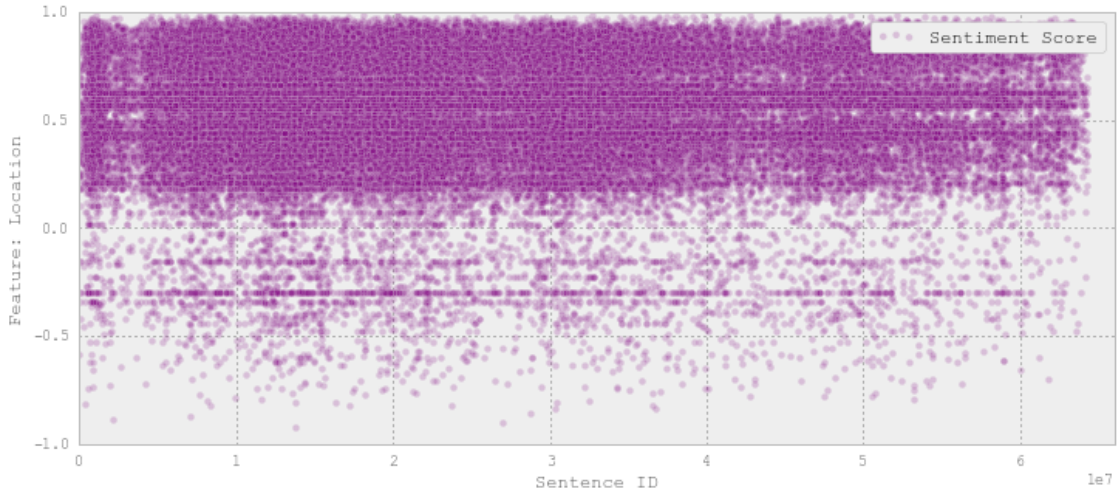
4.4 Communication

```
In [8]: com=file[file['Feature: Communication']!=0]
        com[['Review ID','Feature: Communication']].plot(kind='scatter',
               x='Review ID', y='Feature: Communication', color='Red',
               label='Sentiment Score', alpha=0.2, figsize=(12,5)).set_xlabel("Sent
        plt.axis([0,65999999,-1.0,1.0])
        plt.show()
        comr=com.round(1)
```



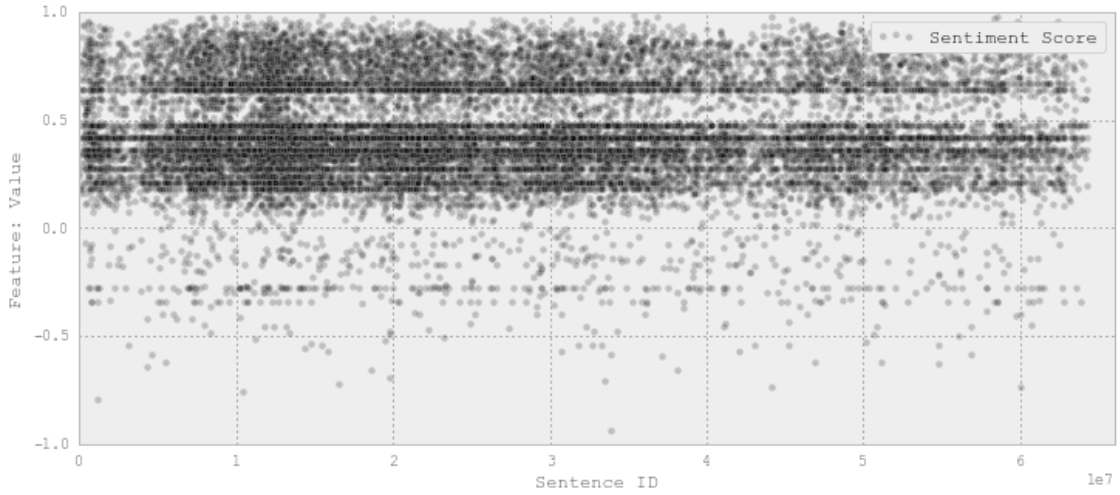
4.5 Location

```
In [9]: loc=file[file['Feature: Location']!=0]
        loc[['Review ID','Feature: Location']].plot(kind='scatter',
              x='Review ID', y='Feature: Location',color='Purple',
              label='Sentiment Score', alpha=0.2, figsize=(12,5)).set_xlabel("Sente
        plt.axis([0,65999999,-1.0,1.0])
        plt.show()
        locr=loc.round(1)
```



4.6 Value

```
In [10]: val=file[file['Feature: Value']!=0]
         val[['Review ID','Feature: Value']].plot(kind='scatter', x='Review ID',
                y='Feature: Value', color='Black',label='Sentiment Score', alpha=0.2
               figsize=(12,5)).set_xlabel("Sentence ID")
         plt.axis([0,65999999,-1.0,1.0])
         plt.show()
         valr=val.round(1)
```

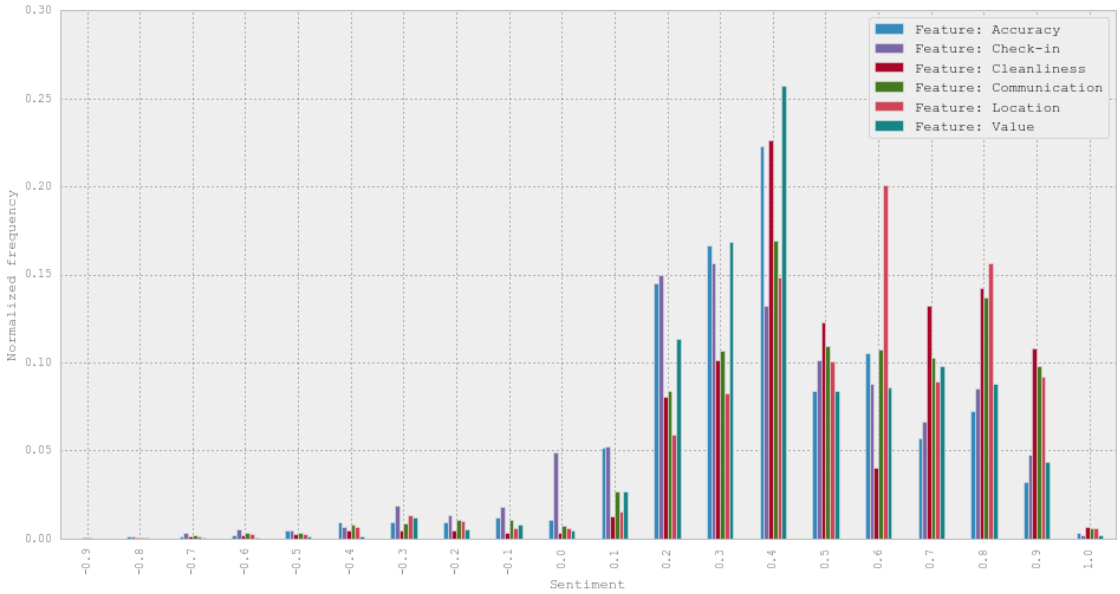


5 Normalized frequencies of sentiment scores per feature (compared)

For plotting the normalized frequencies, the chosen method is to put all the features in the same graph so we can compare their frequencies. From the graph we can see that: * The sentiment score most frequent for the majority of features is 0.4 * For **Accuracy** the peak values are 0.4 and 0.3 * For **Check-in** are 0.3 and 0.2 * For **Cleanliness** we have mostly 0.4, 0.8 and 0.7 (quite positive ones). * The **Communication** reaches its peak on 0.4 and 0.8 * **Location** on 0.6 and 0.8 (just like the frequency of all sentences) * And lastly **Value** reaches the peak on 0.4 and 0.3

From this comparison we can say that the most negative feature in general appears to be **Check-in** and the most positive one **Location**.

```
In [11]: f1=acr['Feature: Accuracy'].value_counts(normalize=True, sort=False)
         f2=chkr['Feature: Check-in'].value_counts(normalize=True, sort=False)
         f3=cler['Feature: Cleanliness'].value_counts(normalize=True, sort=False)
         f4=comr['Feature: Communication'].value_counts(normalize=True, sort=False)
         f5=locr['Feature: Location'].value_counts(normalize=True, sort=False)
         f6=valr['Feature: Value'].value_counts(normalize=True, sort=False)
         frames=[f1,f2,f3,f4,f5,f6]
         result=pd.concat(frames,axis=1)
         result['Sentiment']=result.index
         result.sort_values('Sentiment', axis=0, ascending=True)
         result.plot.bar(x='Sentiment',
           y=['Feature: Accuracy','Feature: Check-in', 'Feature: Cleanliness',
            'Feature: Communication','Feature: Location', 'Feature: Value'],
               figsize=(16,8)).set_ylabel("Normalized frequency");
```



5.1 Mean and stardard deviation of sentiment scores per sentence

The mean sentiment score of all sentences is 0.47 and their standard deviation is 0.336. This means that most of the values fall into 1 standard deviation above and below the mean. The mean itself represent a mid positive sentiment score.

```
In [12]: # Mean and standard deviation of this distribution
         std=ak['Sentiment score'].std()
         std.round(3)
         m=ak['Sentiment score'].mean()
         print 'Mean: ', m.round(3)
         print 'Standard deviation: ', std.round(3)
```

```
Mean:  0.47
Standard deviation:  0.336
```

5.2 Number of sentences with negative sentiment

In total we have **12 798** sentences with negative sentiment, meaning **4.2%** of all sentences are negative. This a very low amount.

```
In [21]: # Select the rows with negative sentiment
         nr_sentences=file['Sentiment score'].count()
         negative_sentences = file[file['Sentiment score'] < 0]
         neg=negative_sentences['Listing ID'].value_counts().sum()
         print neg
         print ((neg/nr_sentences)*100).round(1),'%'
```

```
 12798
4.2 %
```

5.3 Number of sentences with positive sentiment

The number of postive sentences is **234 904**, meaning **78.4%** of all sentences. The rest have no sentiment, so 18.4% are sentences with 0 sentiment.

```
In [22]: # Select the rows with positive sentiment
         pos_sentences = file[file['Sentiment score'] > 0]
         pos=pos_sentences['Listing ID'].value_counts().sum()
         print 'Number of sentences with positive sentiment: ',pos
         print ((pos/nr_sentences)*100).round(1),'%'
```

```
Number of sentences with positive sentiment:  236904
78.4 %
```

5.4 Comparison of negative/positive sentences

From the comparison of positive and negative sentences, we see that the number of positive ones is 18.5 times higher than the negatives.

```
In [16]: print (pos/neg).round(2)

18.51
```

5.5 Percentage of sentences with sentiment 0

There are **52 379** sentences with no sentiment, meaning 17.34% of the total sentences.

```
In [23]: sentences_zero=file[file['Sentiment score']==0]
         nrs=sentences_zero['Sentence'].count()
         print 'Number of sentences with sentiment zero: ', nrs
         print 'Percentage: ',((nrs/nr_sentences)*100).round(2),'%'

Number of sentences with sentiment zero:  52379
Percentage:  17.34 %
```

5.6 Number of automatic message of cancellation

We count the times that the system generates a cancellation message. So in **3 777** sentences we have a cancellation message.

```
In [24]: cancelled=file[file['Sentence']=='This is an automated posting.']
         can=cancelled['Sentence'].count()*3
         # because this message comes in three sentences with all get sentiment 0
         print 'Number of sentences about cancellation: ', can

Number of sentences about cancellation:  3777
```

5.7 Sentences with zero sentiment from reviews

Thus, sentences with no sentiment actually are 16.09%

```
In [25]: print 'Number of sentences with sentiment 0 from pipeline: ', nrs-can
         print 'Percentage: ',(((nrs-can)/nr_sentences)*100).round(2),'%'

Number of sentences with sentiment 0 from pipeline:  48602
Percentage:  16.09 %
```

5.8 Which listing have cancelled most

ID of Listings who have cancelled most (have an automatic posting message). We can see that the highest number of cancellation is for listing with ID 1808512, and it counts 27 times cancelling.

```
In [26]: a=cancelled[['Review ID','Listing ID']].groupby('Listing ID').count()
         a.columns=['# of cancellations']
         a['Listing ID']=a.index
         a.sort_values('# of cancellations', axis=0, ascending=False)[:5] ## Top 5
```

```
Out[26]:             # of cancellations  Listing ID
         Listing ID
         1808512                     27     1808512
         1959016                     16     1959016
         2815820                     15     2815820
         2040532                     14     2040532
         1199315                     13     1199315
```

5.9 Number of listings which have cancelled at least once

We found out that **32.8%** of the listings have cancelled at least once the reservation.

```
In [30]: ten=file.groupby('Listing ID')
         total=ten['Review ID'].mean().count()
         total

         nrlis=a['Listing ID'].count()
         print nrlis, 'listings'
         print ((nrlis/total)*100).round(2),'%'
```

```
773 listings
32.81 %
```

5.10 Average number of cancellations per listing

In average, the number of cancellations per listing is **1.63**

```
In [28]: avg=a['# of cancellations'].mean()
         print avg.round(2),'Rounded: ',avg.round(0)
```

```
1.63 Rounded:  2.0
```