Data_Analysis

August 28, 2017

1 Amazon Fine Food Reviews: EDA

1.1 Introduction

Data Fields Explanation

The Amazon Fine Food Reviews dataset consists of 568,454 food reviews. This dataset consists of a single CSV file, Reviews.csv. The columns in the table are:

```
In [63]: #from IPython.display import Image
         #Image(filename='AmazonReview.png')
```

1.2 Objective

Analysing the data & plot the required graphs to show that these conclusions are true:

Note: This notebook is highly inspired from the Exploratory visualization of Amazon fine food reviews by Rob Castellano.

1.3 Loading the Data

```
In [64]: #Let's import pandas to read the csv file.
         import pandas as pd
         df = pd.read_csv("Reviews.csv")
```

```
In [65]: #Printing first 5 columns from our data frame
         df.head()
```

```
Out[65]:    Id   ProductId          UserId                      ProfileName  \
         0   1  B001E4KFG0  A3SGXH7AUHU8GW                       delmartian
         1   2  B00813GRG4  A1D87F6ZCVE5NK                           dll pa
         2   3  B000LQOCH0   ABXLMWJIXXAIN  Natalia Corres "Natalia Corres"
         3   4  B000UA0QIQ  A395BORC6FGVXV                             Karl
         4   5  B006K2ZZ7K  A1UQRSCLF8GW1T    Michael D. Bigham "M. Wassir"

            HelpfulnessNumerator  HelpfulnessDenominator  Score        Time  \
         0                     1                       1      5  1303862400
         1                     0                       0      1  1346976000
         2                     1                       1      4  1219017600
         3                     3                       3      2  1307923200
```

```
   4                      0                        0      5  1350777600

                  Summary                                               Text
   0  Good Quality Dog Food  I have bought several of the Vitality canned d...
   1      Not as Advertised  Product arrived labeled as Jumbo Salted Peanut...
   2  "Delight" says it all  This is a confection that has been around a fe...
   3         Cough Medicine  If you are looking for the secret ingredient i...
   4            Great taffy  Great taffy at a great price.  There was a wid...
```

In [66]:
```python
#Observing the lables of each column
print(df.keys())
```

```
Index([u'Id', u'ProductId', u'UserId', u'ProfileName', u'HelpfulnessNumerator',
       u'HelpfulnessDenominator', u'Score', u'Time', u'Summary', u'Text'],
      dtype='object')
```

In [67]:
```python
#Observing the shape of our data frame.
df.shape
# Note: We have 10 features and 568454 data points.
```

Out[67]:
```
(568454, 10)
```

In [68]:
```python
#Lets check for missing values
df.info()
#Observe that there are some missing values in 'PROFILENAME' & 'SUMMARY' column.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
Id                        568454 non-null int64
ProductId                 568454 non-null object
UserId                    568454 non-null object
ProfileName               568438 non-null object
HelpfulnessNumerator      568454 non-null int64
HelpfulnessDenominator    568454 non-null int64
Score                     568454 non-null int64
Time                      568454 non-null int64
Summary                   568428 non-null object
Text                      568454 non-null object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
```

In [69]:
```python
df.describe()
#Observe that more than 75% of our data is belonging to positive /
#class, i.e. we have imbalanced dataset.
```

Out[69]:
```
                  Id  HelpfulnessNumerator  HelpfulnessDenominator  \
count  568454.000000         568454.000000            568454.00000
```

```
mean    284227.500000                 1.743817                 2.22881
std     164098.679298                 7.636513                 8.28974
min          1.000000                 0.000000                 0.00000
25%     142114.250000                 0.000000                 0.00000
50%     284227.500000                 0.000000                 1.00000
75%     426340.750000                 2.000000                 2.00000
max     568454.000000               866.000000               923.00000

                Score          Time
count  568454.000000  5.684540e+05
mean        4.183199  1.296257e+09
std         1.310436  4.804331e+07
min         1.000000  9.393408e+08
25%         4.000000  1.271290e+09
50%         5.000000  1.311120e+09
75%         5.000000  1.332720e+09
max         5.000000  1.351210e+09
```

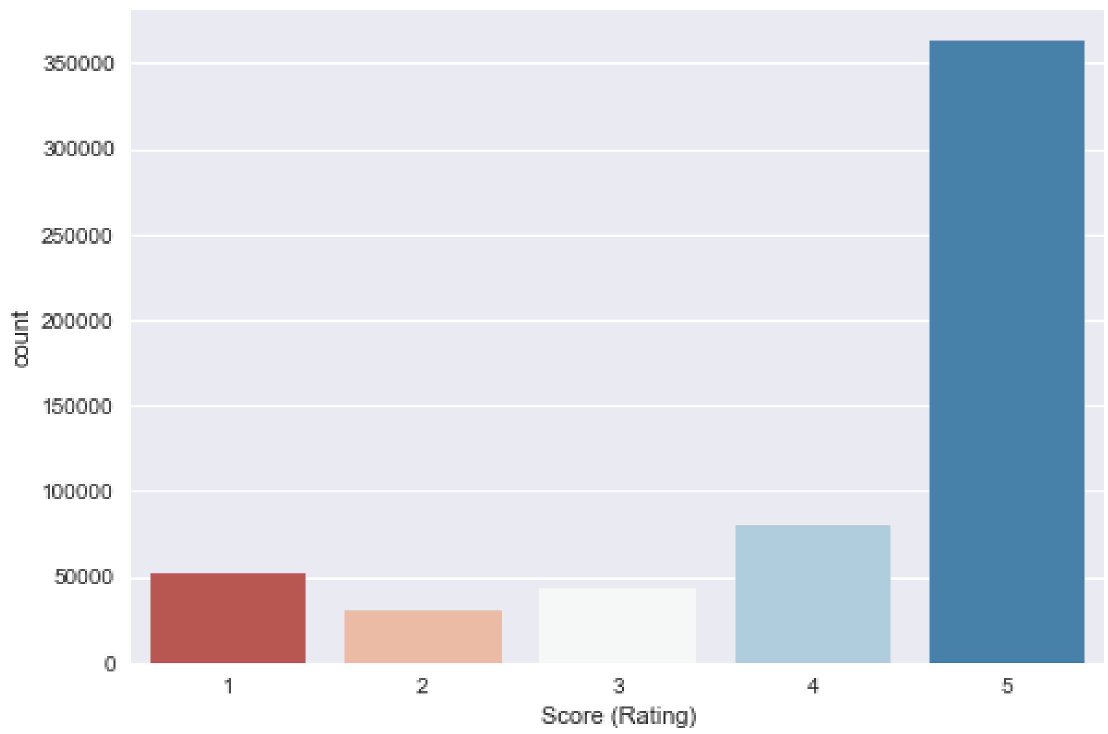
In [70]:
```python
#Lets do the value count on 'Scores'.
df.Score.value_counts()
```

Out[70]:
```
5    363122
4     80655
1     52268
3     42640
2     29769
Name: Score, dtype: int64
```

1.4 Exploratory Data Analysis

Till now we saw that 5-star reviews constitute a large proportion (64%) of all reviews. The next most prevalent rating is 4-stars(14%), followed by 1-star (9%), 3-star (8%), and finally 2-star reviews (5%). Note that we have 10 features and 568454 data points. There are some missing values in 'PROFILENAME' & 'SUMMARY' column. More than 75% of our data is belonging to positive class(Score=4,5), i.e. we have imbalanced dataset.

In [71]:
```python
#Importing Seaborn and Matplotlib for graphical effects.
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure()
sns.countplot(x='Score', data=df, palette='RdBu')
plt.xlabel('Score (Rating)')
plt.show()
```



1.5 Creating a new dataframe

In [72]:
```
#copying the original dataframe to 'temp_df'.
temp_df = df[['UserId','HelpfulnessNumerator','HelpfulnessDenominator', 'Summary', 'Te

#Adding new features to dataframe.
temp_df["Sentiment"] = temp_df["Score"].apply(lambda score: "positive" if score > 3 el
                                               ("negative" if score < 3 else "not defi
temp_df["Usefulness"] = (temp_df["HelpfulnessNumerator"]/temp_df["HelpfulnessDenominat
(lambda n: ">75%" if n > 0.75 else ("<25%" if n < 0.25 else ("25-75%" if n >= 0.25 and
                                                                         n <= 0.75 else


temp_df.loc[temp_df.HelpfulnessDenominator == 0, 'Usefulness'] = ["useless"]
# Removing all rows where 'Score' is equal to 3
#temp_df = temp_df[temp_df.Score != 3]
#Lets now observe the shape of our new dataframe.
temp_df.shape
```

Out[72]: (568454, 8)

In [73]:
```
temp_df.describe()
```

Out[73]:

	HelpfulnessNumerator	HelpfulnessDenominator	Score
count	568454.000000	568454.00000	568454.000000

```
mean                1.743817                 2.22881       4.183199
std                 7.636513                 8.28974       1.310436
min                 0.000000                 0.00000       1.000000
25%                 0.000000                 0.00000       4.000000
50%                 0.000000                 1.00000       5.000000
75%                 2.000000                 2.00000       5.000000
max               866.000000               923.00000       5.000000
```

In [74]:
```
temp_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 8 columns):
UserId                    568454 non-null object
HelpfulnessNumerator      568454 non-null int64
HelpfulnessDenominator    568454 non-null int64
Summary                   568428 non-null object
Text                      568454 non-null object
Score                     568454 non-null int64
Sentiment                 568454 non-null object
Usefulness                568454 non-null object
dtypes: int64(3), object(5)
memory usage: 34.7+ MB
```

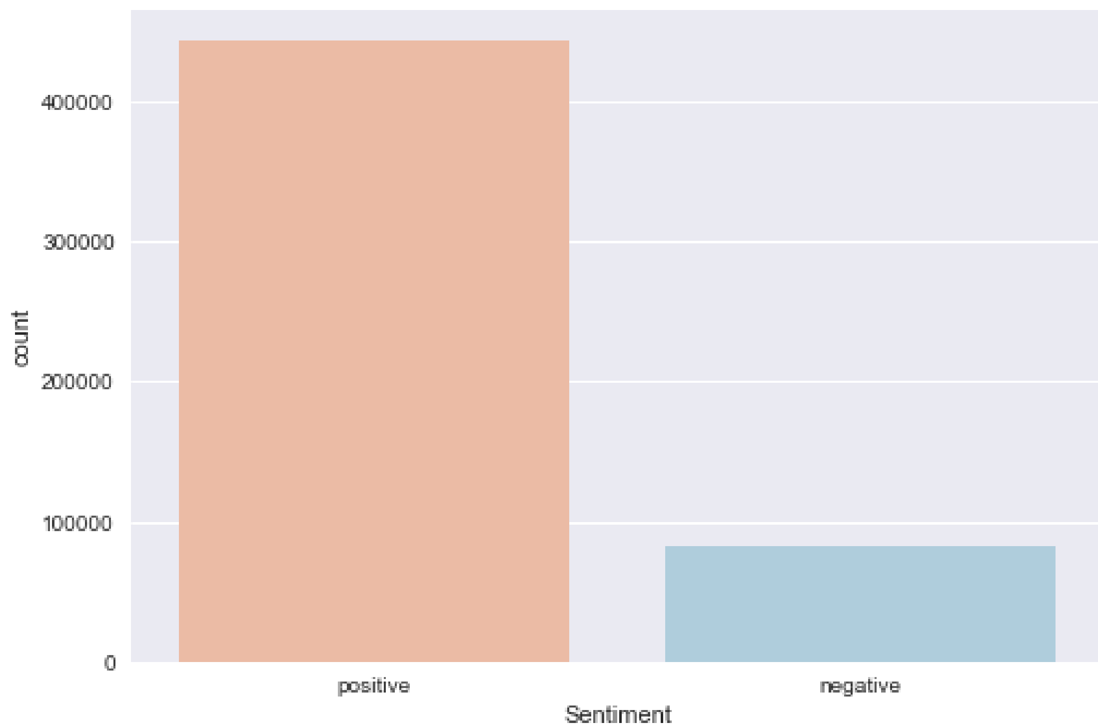
In [75]:
```
#Lets view the dataframe when Score=5
temp_df[temp_df.Score == 5].head(10)
```

Out[75]:
```
            UserId  HelpfulnessNumerator  HelpfulnessDenominator  \
0   A3SGXH7AUHU8GW                     1                       1
4   A1UQRSCLF8GW1T                     0                       0
6   A1SP2KVKFXXRU1                     0                       0
7   A3JRGQVEQN31IQ                     0                       0
8   A1MZYO9TZK0BBI                     1                       1
9   A21BT40VZCCYT4                     0                       0
10  A3HDKO7OW0QNK4                     1                       1
11  A2725IB4YY9JEB                     4                       4
14  A2MUGFV2TDQ47K                     4                       5
15  A1CZX3CP8IKQIJ                     4                       5

                                              Summary  \
0                               Good Quality Dog Food
4                                         Great taffy
6        Great!  Just as good as the expensive brands!
7                             Wonderful, tasty taffy
8                                          Yay Barley
9                                    Healthy Dog Food
10                        The Best Hot Sauce in the World
11  My cats LOVE this "diet" food better than thei...
```

```
14                      Strawberry Twizzlers - Yummy
15            Lots of twizzlers, just what you expect.

                                                 Text  Score Sentiment  \
0   I have bought several of the Vitality canned d...      5  positive
4   Great taffy at a great price.  There was a wid...      5  positive
6   This saltwater taffy had great flavors and was...      5  positive
7   This taffy is so good.  It is very soft and ch...      5  positive
8   Right now I'm mostly just sprouting this so my...      5  positive
9   This is a very healthy dog food. Good for thei...      5  positive
10  I don't know if it's the cactus or the tequila...      5  positive
11  One of my boys needed to lose some weight and ...      5  positive
14  The Strawberry Twizzlers are my guilty pleasur...      5  positive
15  My daughter loves twizzlers and this shipment ...      5  positive

   Usefulness
0        >75%
4     useless
6     useless
7     useless
8        >75%
9     useless
10       >75%
11       >75%
14       >75%
15       >75%
```

1.6 Positive reviews are very common

In [76]:
```
sns.countplot(x='Sentiment', order=["positive", "negative"], data=temp_df, palette='Ro
plt.xlabel('Sentiment')
plt.show()
```




```
In [77]: temp_df.Sentiment.value_counts()
```

```
Out[77]: positive       443777
         negative        82037
         not defined     42640
         Name: Sentiment, dtype: int64
```

Therefore we could conclude that the positive reviews are way more than the negative reviews.

1.7 Polpular words in Review

A look at the post popular words in positive (4-5 stars) and negative (1-2 stars) reviews shows that both positive and negative reviews share many popular words, such as "coffee", "taste", "flavor", "price", "good", and "product." The words "good", "great", "love", "favorite", and "find" are indicative of positive reviews, while negative reviews contain words such as "didn't" and "disappointed", but these distinguishing words appear less frequently than distinguishing words in positive reviews.

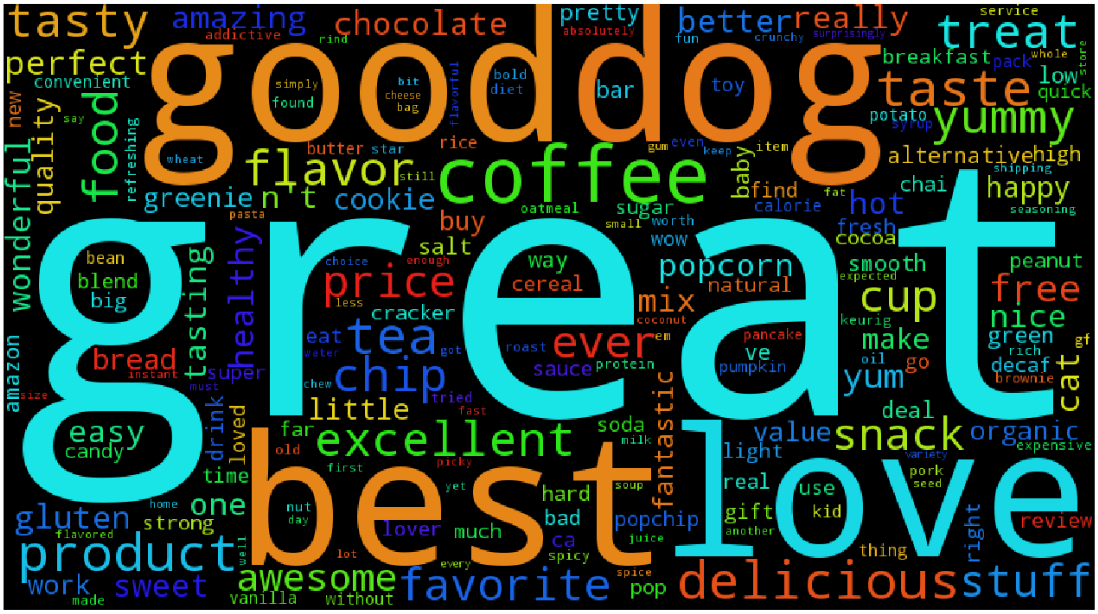
```
In [78]: pos = temp_df.loc[temp_df['Sentiment'] == 'positive']
         pos = pos[0:25000]

         neg = temp_df.loc[temp_df['Sentiment'] == 'negative']
         neg = neg[0:25000]
```

In [79]:
```
import nltk
from nltk.corpus import stopwords
from wordcloud import WordCloud
import string
import matplotlib.pyplot as plt


def create_Word_Corpus(temp):
    words_corpus = ''
    for val in temp["Summary"]:
        text = str(val).lower()
        #text = text.translate(trantab)
        tokens = nltk.word_tokenize(text)
        tokens = [word for word in tokens if word not in stopwords.words('english')]
        for words in tokens:
            words_corpus = words_corpus + words + ' '
    return words_corpus

# Generate a word cloud image
pos_wordcloud = WordCloud(width=900, height=500).generate(create_Word_Corpus(pos))
neg_wordcloud = WordCloud(width=900, height=500).generate(create_Word_Corpus(neg))
```

```
C:\Users\Kanav\Anaconda2\lib\site-packages\ipykernel_launcher.py:14: UnicodeWarning: Unicode e
```

In [80]:
```
# Plot cloud
def plot_Cloud(wordCloud):
    plt.figure( figsize=(20,10), facecolor='w')
    plt.imshow(wordCloud)
    plt.axis("off")
    plt.tight_layout(pad=0)
    plt.show()
    plt.savefig('wordclouds.png', facecolor='w', bbox_inches='tight')
```

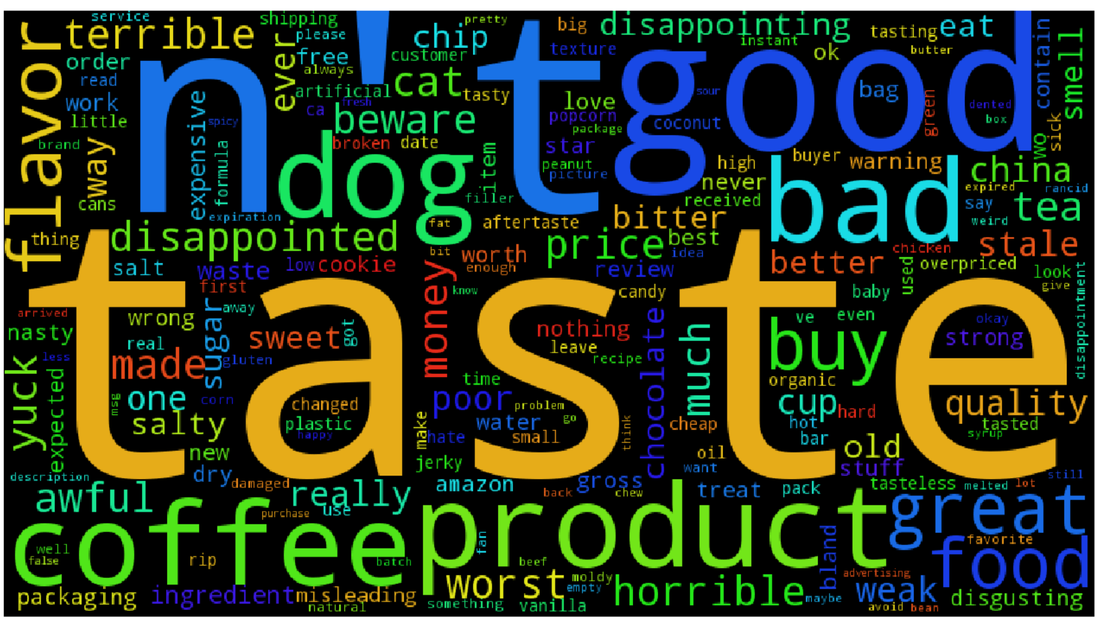
In [81]:
```
#Visuallizing popular positive words
plot_Cloud(pos_wordcloud)
```



In [82]: *#Visuallizing popular negative words*

```
plot_Cloud(neg_wordcloud)
```

<matplotlib.figure.Figure at 0x4b9b4860>



1.8 Helpfulness

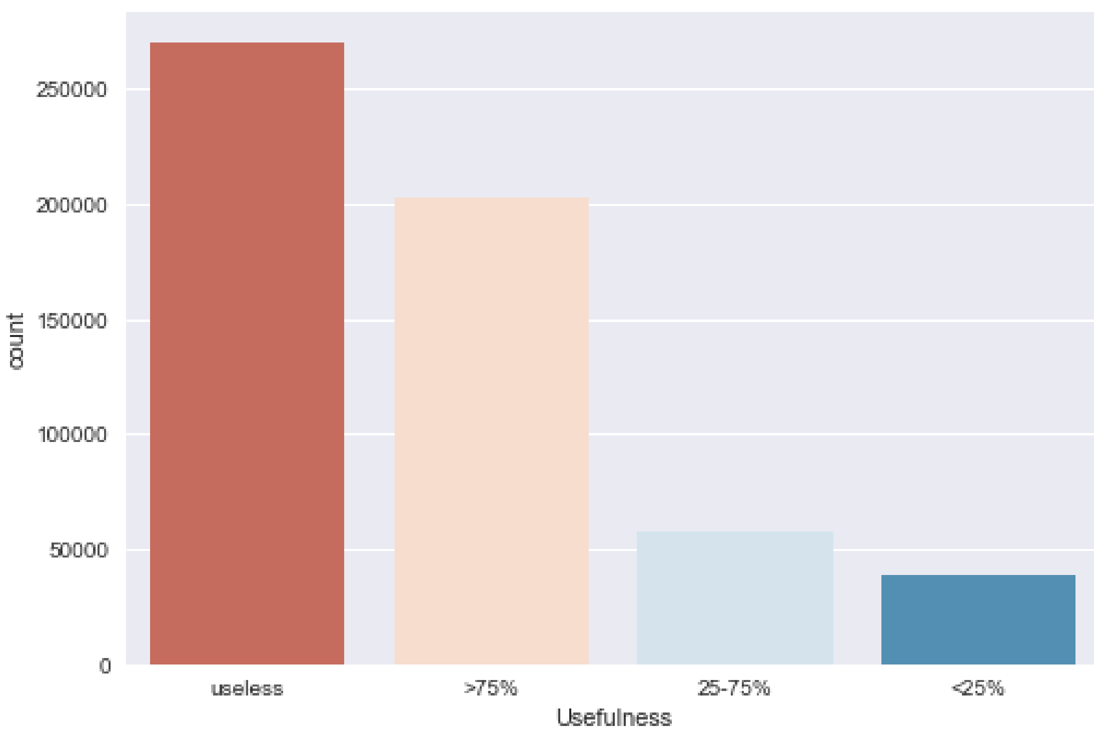
1.8.1 How many reviews are helpful?

Among all reviews, almost half (50%) are not voted on at all. Among reviews that are voted on, helpful reviews(>75%) are the most common

In [83]:
```
#Checking the value count for 'Usefulness'
temp_df.Usefulness.value_counts()
```

Out[83]:
```
useless    270052
>75%       202836
25-75%      57286
<25%        38280
Name: Usefulness, dtype: int64
```

In [84]:
```
sns.countplot(x='Usefulness', order=['useless', '>75%', '25-75%', '<25%'], data=temp_d
plt.xlabel('Usefulness')
plt.show()
```



1.8.2 Positive reviews are found more helpful

As the rating becomes more positive, the reviews become more helpful (and less unhelpful).

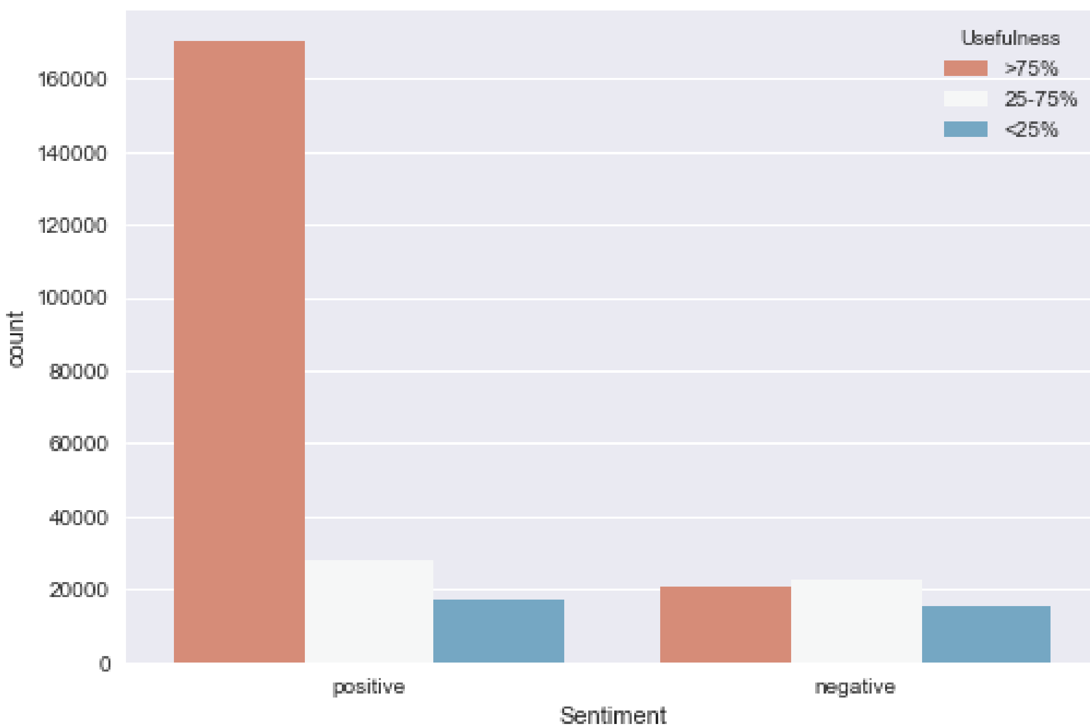
In [85]:
```
temp_df[temp_df.Score==5].Usefulness.value_counts()
```

```
Out[85]: useless    186743
         >75%       142954
         25-75%      21314
         <25%        12111
         Name: Usefulness, dtype: int64
```

```
In [86]: temp_df[temp_df.Score==2].Usefulness.value_counts()
```

```
Out[86]: useless    10604
         >75%        7423
         25-75%      6693
         <25%        5049
         Name: Usefulness, dtype: int64
```

```
In [87]: sns.countplot(x='Sentiment', hue='Usefulness', order=["positive", "negative"], \
                       hue_order=['>75%', '25-75%', '<25%'], data=temp_df, palette='RdBu')
         plt.xlabel('Sentiment')
         plt.show()
```



Therefore positive reviews are more helpful.

1.9 Word Count

```
In [88]: temp_df["text_word_count"] = temp_df["Text"].apply(lambda text: len(text.split()))
```

In [89]:
```python
temp_df.head()
```

Out[89]:
```
              UserId  HelpfulnessNumerator  HelpfulnessDenominator  \
0     A3SGXH7AUHU8GW                     1                       1
1     A1D87F6ZCVE5NK                     0                       0
2      ABXLMWJIXXAIN                     1                       1
3     A395BORC6FGVXV                     3                       3
4     A1UQRSCLF8GW1T                     0                       0

                 Summary                                               Text  \
0  Good Quality Dog Food  I have bought several of the Vitality canned d...
1      Not as Advertised  Product arrived labeled as Jumbo Salted Peanut...
2  "Delight" says it all  This is a confection that has been around a fe...
3         Cough Medicine  If you are looking for the secret ingredient i...
4            Great taffy  Great taffy at a great price.  There was a wid...

   Score Sentiment Usefulness  text_word_count
0      5  positive       >75%               48
1      1  negative    useless               31
2      4  positive       >75%               94
3      2  negative       >75%               41
4      5  positive    useless               27
```

In [90]:
```python
temp_df[temp_df.Score==5].text_word_count.median()
```

Out[90]: 52.0

In [91]:
```python
temp_df[temp_df.Score==4].text_word_count.median()
```

Out[91]: 65.0

In [92]:
```python
temp_df[temp_df.Score==3].text_word_count.median()
```

Out[92]: 70.0

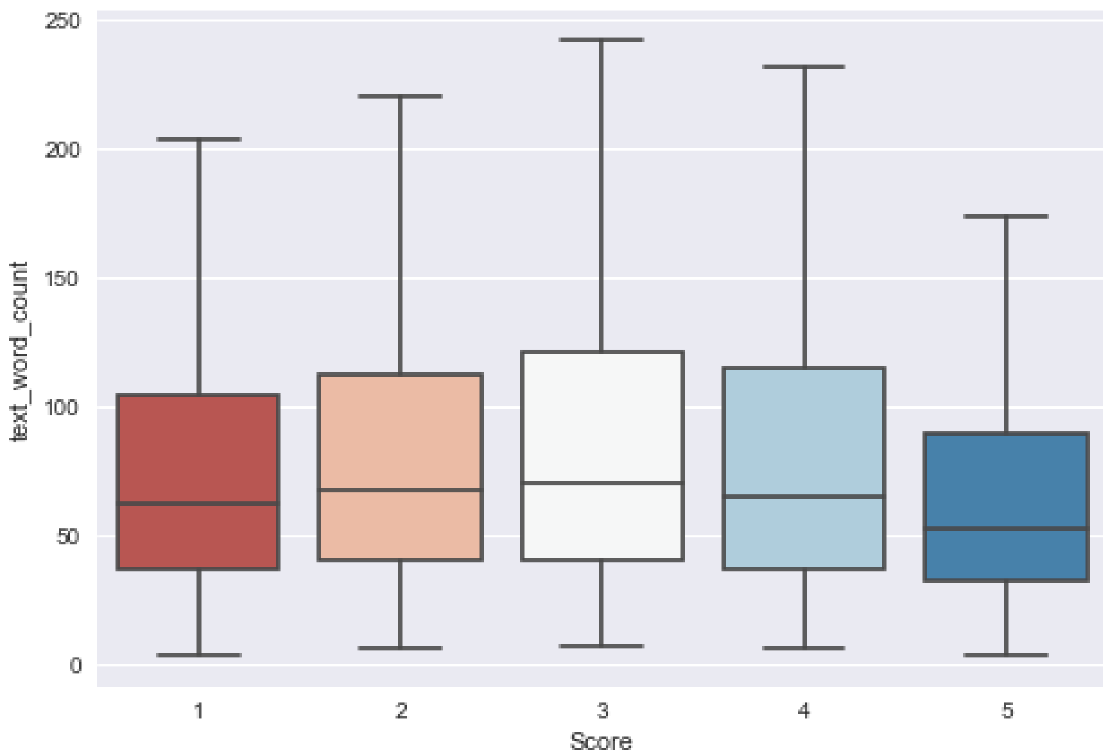
In [93]:
```python
temp_df[temp_df.Score==2].text_word_count.median()
```

Out[93]: 67.0

In [94]:
```python
temp_df[temp_df.Score==1].text_word_count.median()
```

Out[94]: 62.0

In [104]:
```python
sns.boxplot(x='Score',y='text_word_count', data=temp_df, palette='RdBu', showfliers=
plt.show()
```

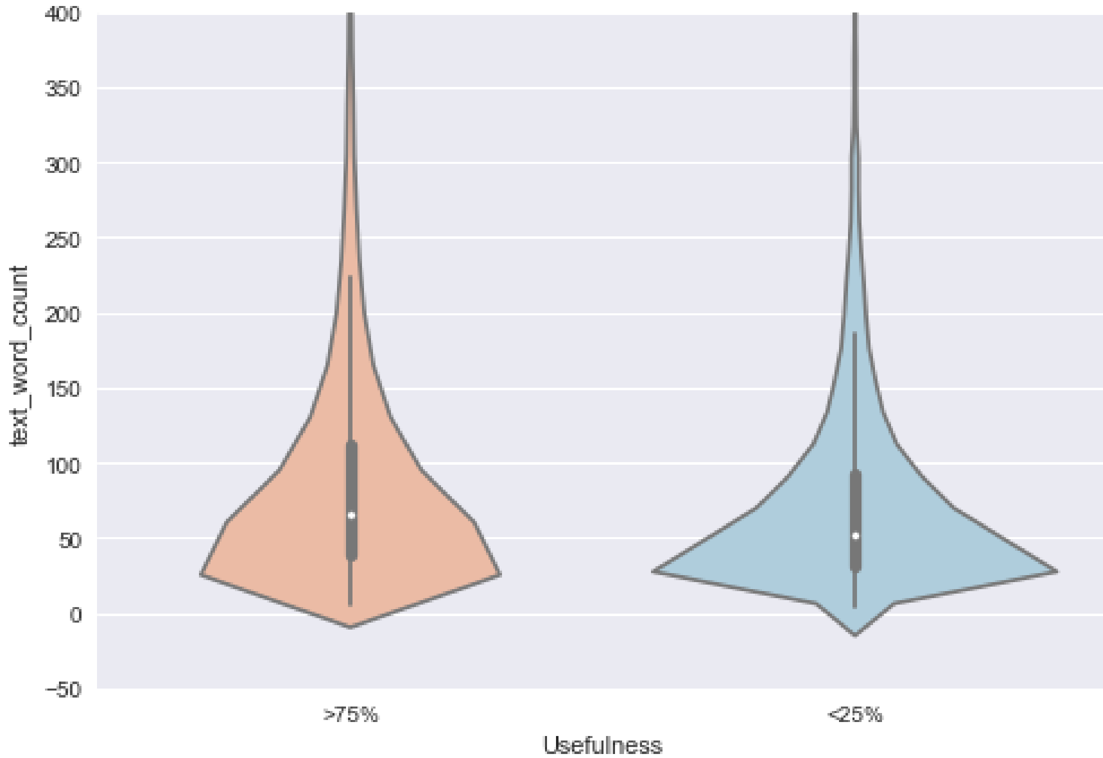


Observations: 5-star reviews had the lowest median word count (52 words), while 3-star reviews had the largest median word count (70 words).

1.9.1 How does word count relate to helpfulness?

The word counts for helpful reviews and not helpful reviews have a similar distribution with the greatest concentration of reviews of approximately 25 words. However, not helpful reviews have a larger concentration of reviews with low word count and helpful reviews have more longer reviews. Helpful reviews have a higher median word count (67 words) than not helpful reviews (54 words).

```
In [105]: sns.violinplot(x='Usefulness', y='text_word_count', order=[">75%", "<25%"], \
                         data=temp_df, palette='RdBu')
          plt.ylim(-50, 400)
          plt.show()
```



1.10 Frequency of reviewers

Using User IDs, one can recognize repeat reviewers. Reviewers that have reviewed over 50 products account for over 5% of all reviews in the database. We will call such reviewers frequent reviewers. (The cutoff choice of 50, as opposed to another choice, seemed to not have a larger impact on the results.) I asked: Does the behavior of frequent reviewers differ from that of infrequent reviewers?

```
In [106]: x = temp_df.UserId.value_counts()
          x.to_dict()
          print("converted Series to dictionary")

converted Series to dictionary
```

```
In [107]: temp_df["reviewer_freq"] = temp_df["UserId"].apply(lambda counts: "Frequent (>50 revi
                                                                         if x[counts]>50 else
```

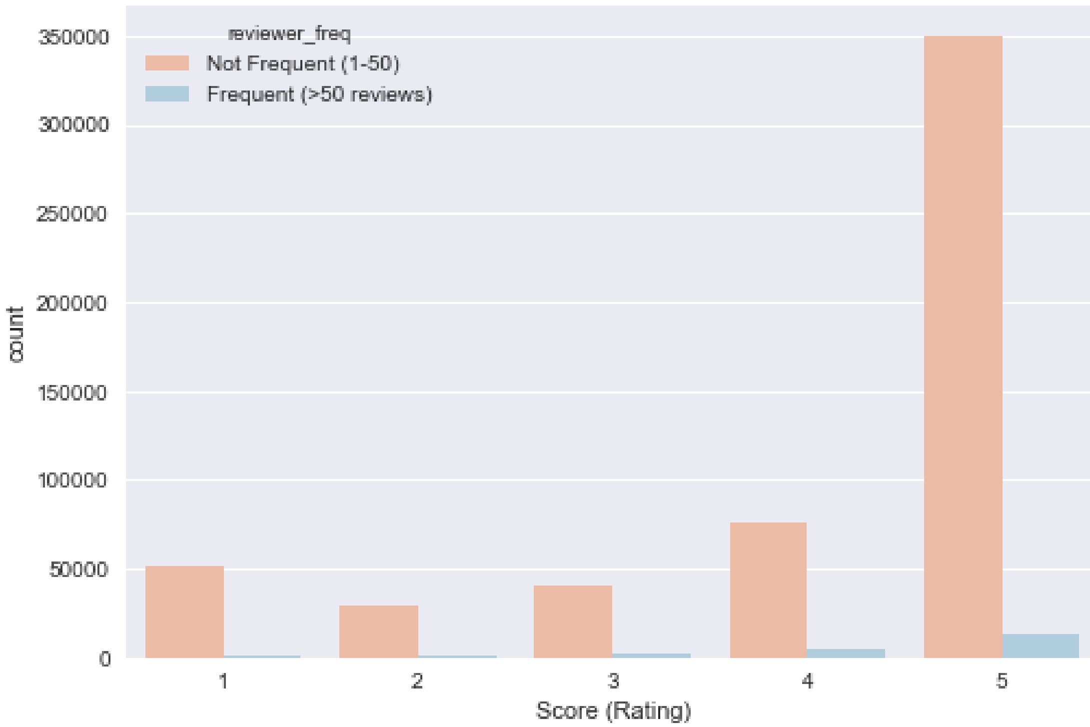
```
In [108]: temp_df.head()

Out[108]:            UserId  HelpfulnessNumerator  HelpfulnessDenominator  \
          0  A3SGXH7AUHU8GW                     1                       1
          1  A1D87F6ZCVE5NK                     0                       0
```

```
2   ABXLMWJIXXAIN                       1                        1
3  A395BORC6FGVXV                       3                        3
4  A1UQRSCLF8GW1T                       0                        0

                Summary                                              Text  \
0  Good Quality Dog Food  I have bought several of the Vitality canned d...
1      Not as Advertised  Product arrived labeled as Jumbo Salted Peanut...
2  "Delight" says it all  This is a confection that has been around a fe...
3         Cough Medicine  If you are looking for the secret ingredient i...
4            Great taffy  Great taffy at a great price.  There was a wid...

   Score Sentiment Usefulness  text_word_count         reviewer_freq
0      5  positive       >75%               48  Not Frequent (1-50)
1      1  negative    useless               31  Not Frequent (1-50)
2      4  positive       >75%               94  Not Frequent (1-50)
3      2  negative       >75%               41  Not Frequent (1-50)
4      5  positive    useless               27  Not Frequent (1-50)
```

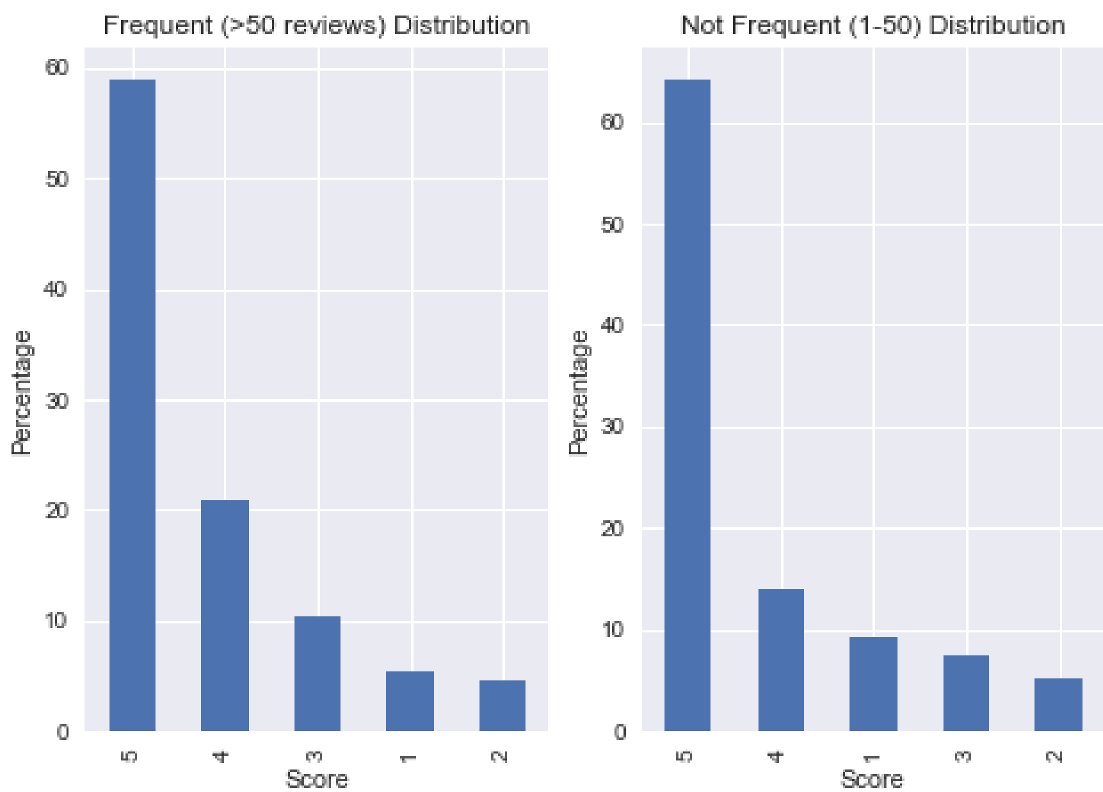
1.10.1 Are frequent reviewers more discerning?

The distribution of ratings among frequent reviewers is similar to that of all reviews. However, we can see that frequent reviewers give less 5-star reviews and less 1-star review. Frequent users appear to be more discerning in the sense that they give less extreme reviews than infrequent reviews.

```
In [109]: ax = sns.countplot(x='Score', hue='reviewer_freq', data=temp_df, palette='RdBu')
          ax.set_xlabel('Score (Rating)')
          plt.show()
```



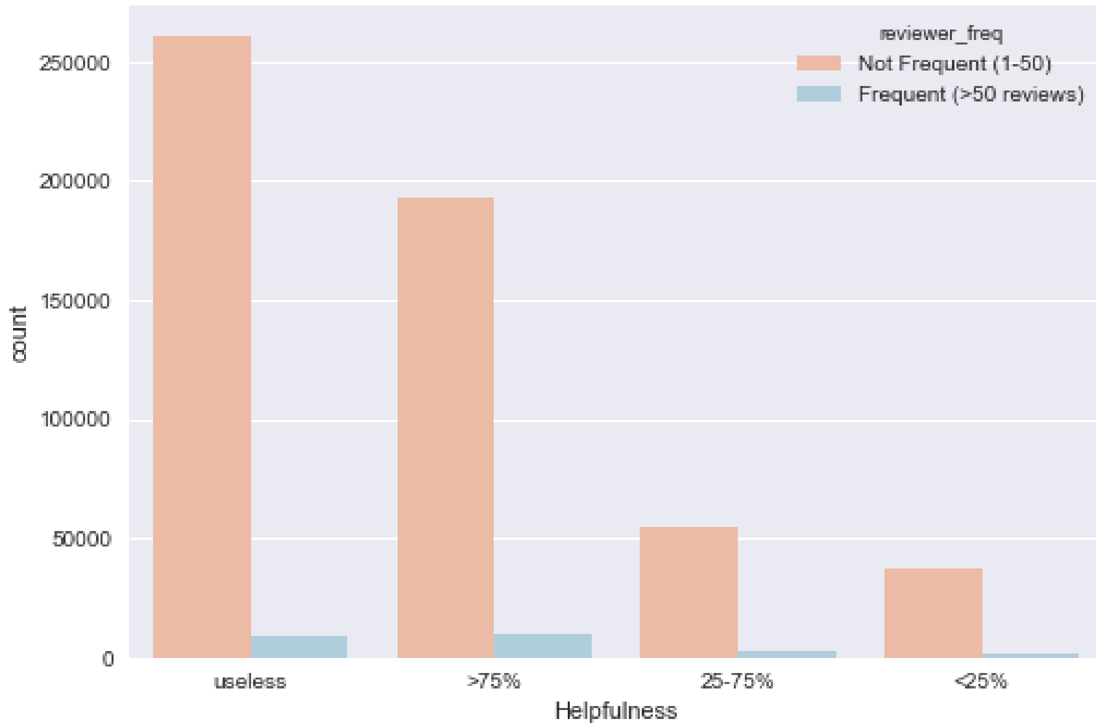
In [110]:
```
y = temp_df[temp_df.reviewer_freq=="Frequent (>50 reviews)"].Score.value_counts()
z = temp_df[temp_df.reviewer_freq=="Not Frequent (1-50)"].Score.value_counts()

tot_y = y.sum()

y = (y/tot_y)*100

tot_z = z.sum()

z = (z/tot_z)*100

ax1 = plt.subplot(121)
y.plot(kind="bar",ax=ax1)
plt.xlabel("Score")
plt.ylabel("Percentage")
plt.title("Frequent (>50 reviews) Distribution")

ax2 = plt.subplot(122)
z.plot(kind="bar",ax=ax2)
plt.xlabel("Score")
plt.ylabel("Percentage")
plt.title("Not Frequent (1-50) Distribution")
plt.show()
```



1.10.2 Are frequent reviewers more helpful?

The distribution of helpfulness for frequent reviewers is similar to that of all reviews. However, frequent reviewers are more likely to have their review voted on and when voted on, more likely to be voted helpful, and less likely to be unhelpful.

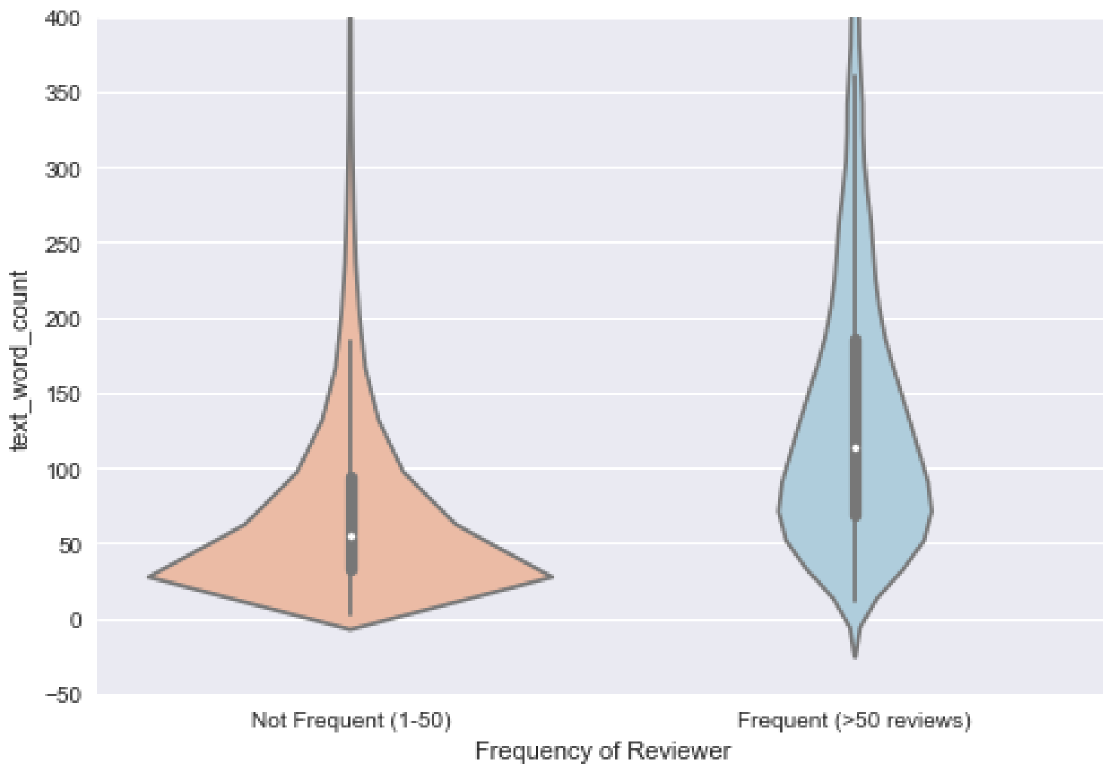
```
In [111]: sns.countplot(x='Usefulness', order=['useless', '>75%', '25-75%', '<25%'], \
                        hue='reviewer_freq', data=temp_df, palette='RdBu')
          plt.xlabel('Helpfulness')
          plt.show()
```



1.10.3 Are frequent reviewers more verbose?

The distributions of word counts for frequent and infrequent reviews shows that infrequent reviewers have a large amount of reviews of low word count. On the other hand, the largest concentration of word count is higher for frequent reviewers than for infrequent reviews. Moreover, the median word count for frequent reviewers is higher than the median for infrequent reviewers.

In [112]:
```
sns.violinplot(x='reviewer_freq', y='text_word_count',  \
               data=temp_df, palette='RdBu')
plt.xlabel('Frequency of Reviewer')
plt.ylim(-50, 400)
plt.show()
```



1.11 Conclusion