# Homework 10

**Name:** Catherine Squillante

**PID:** cat1997

# Is there a relationship between review sentiment and star rating of reviews?

How do we define how good or bad a review is? A review may have a high or low star rating but that does not necessarily define what the reader of the review will take away from it. In order to more accurately measure the sentiment of the review, I will analyze the text of the review. Then, compare this data to the star rating to see if the rating is an accurate depiction of what the customer actually said in their review.

Step 1: load the data from yelp and dictonaries of common positive and negative words

In [93]:

```
import pandas
import numpy
import matplotlib
#get data
data = pandas.read_csv('Homework10.csv')
#get list of good and bad words
positive = open('opinion-lexicon-English/positive-words.txt', "r")
negative = open('opinion-lexicon-English/negative-word.txt', encoding = "ISO-8859-
1")
pos_words = [word.strip('\n') for word in positive.readlines()]
neg_words = [word.strip('\n') for word in negative.readlines()]
positive.close()
negative.close()
negatives = ['disappointing', 'greasy', 'disappointing', 'dry', 'expensive', 'medio
cre', 'hassle']
for word in negatives:
    neg_words.append(word)
```

Step 2: In order to quantify the sentiment of the review text, each review is transformed into a bag of words. The number of occurences of good and bad words in each review is counted. It is then classified as either positive, negative, or neutral. The strength of the sentiment is quamntified by how many more good than bad words or bad than good words the review had.

In [114]:

```
data['length'] = data.Text.map(len)
data['bag'] = data.Text.map(lambda t: t.rstrip('.,?!\n').lower().split())
def find_sentiment(row):
    neg_count = 0
    pos_count = 0
    for word in row:
        word = word.rstrip('.,?!\n')
        if word in pos_words:
            pos_count = pos_count + 1
        if word in neg_words:
            neg_count = neg_count + 1
    if pos_count > neg_count:
        overall = ('Positive', pos_count - neg_count)
    elif neg_count > pos_count:
        overall = ('Negative', neg_count - pos_count)
    else:
        overall = ('Neutral', 0)
    return [pos_count, neg_count, overall]
data['positive_count'] = data.bag.map(lambda t: find_sentiment(t)[0])
data['negative_count'] = data.bag.map(lambda t: find_sentiment(t)[1])
data['sentiment'] = data.bag.map(lambda t: find_sentiment(t)[2][0])
data['sentiment_strength'] = data.bag.map(lambda t: find_sentiment(t)[2][1])
```

Sentiment analysis results:

In [139]:

```
data.iloc[:, 7:].head()
```

Out[139]:

| | bag | positive_count | negative_count | sentiment | sentiment_strength |
|---|---|---|---|---|---|
| **0** | [the, tortilla, chips, were, good,, the, salsa... | 9 | 1 | Positive | 8 |
| **1** | [nice, variety, of, appetizers, and, entree's.... | 9 | 0 | Positive | 9 |
| **2** | [living, in, blacksburg, for, the, past, 3, ye... | 10 | 0 | Positive | 10 |
| **3** | [this, is, easily, my, favorite, restaurant, i... | 9 | 0 | Positive | 9 |
| **4** | [tacos!, there, is, a, good, reason, "taco", i... | 10 | 2 | Positive | 8 |

In [124]:

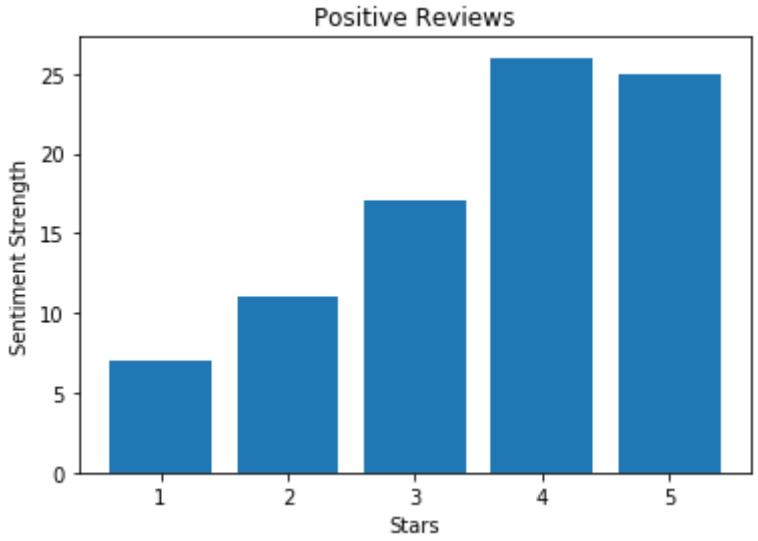
```
positive_data = data.loc[data.sentiment == 'Positive']
negative_data = data.loc[data.sentiment == 'Negative']
```

The graphs below show there is clearly a relationship between the text sentiment and stars. The more strongly positive reviews had consistently more stars, while the strongly negative reviews had consistently low amount of stars. Although there is clearly a correlation between sentiment and star ratings, shows that there is some disconnection between the two. For example, there are still many reviews that had a positive text sentiment but were given a low amount of stars by the customer. This suggests that either the customer's personal ratings aren't always in line with what they say, or that the sentiment word analysis failed to accurately describe the text

In [141]:

```
pos = plot.bar(positive_data.Stars, positive_data.sentiment_strength, )
plot.xlabel('Stars')
plot.ylabel('Sentiment Strength')
plot.title('Positive Reviews')
display(pos)
```

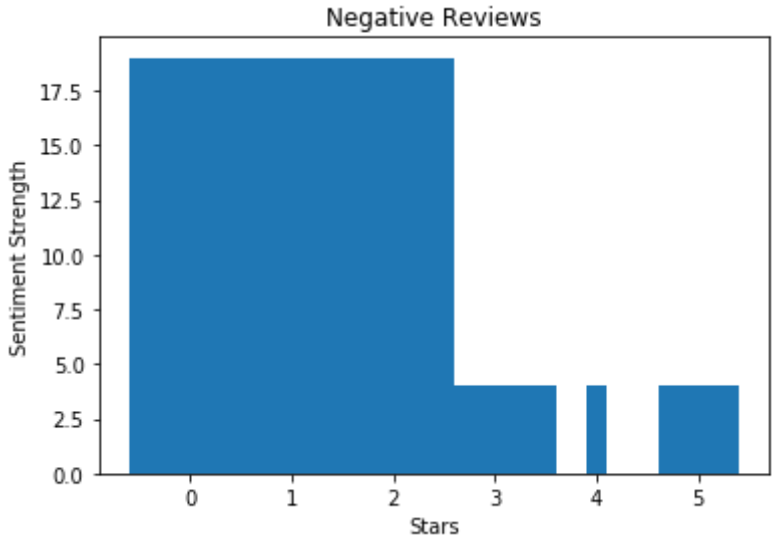
```
<BarContainer object of 1438 artists>
```



In [142]:

```
neg = plot.bar(negative_data.Stars, negative_data.sentiment_strength)
plot.xlabel('Stars')
plot.ylabel('Sentiment Strength')
plot.title('Negative Reviews')
display(neg)
```

```
<BarContainer object of 124 artists>
```




# Conclusion

The data shows that there is some disconnect between sentiment of review text and actual star. This is important because despite star rating, the sentiment of the review could play a role in future customer's decisions. In order to get an accurate depiction of how the reviews are going to influence readers, the sentiment of the review text must be taken into account. Some suggestions on how to make these two attributes more closely line up is that yelp could give the reviewers some prompts or suggestions on what to write based on the star rating of their review, such as asking them to describe what was so good or bad specifically about the experience using certain key words that have a connotation that aligns with their star rating.