Homework 4

Wenjun Zhang
A53218995

Problems:

1. Use the first 5000 reviews in the "50,000 beer reviews" data, the number of unique bigrams and the 5 most-frequently-occurring bigrams are:

```
There are  182246 unique bigrams
The 5 most frequently occuring bigrams are ['with-a', 'in-the', 'of-the', 'is-a', 'on-the']
```

---

2. Adapt the provided code to use the 1000 most common bigrams, the MSE is:

```
The MSE is  0.343153014061
```

3. Repeat the above experiment using unigrams and bigrams, still considering the 1000 most common. The MSE obtained using the new predictor is:

```
The MSE is  0.289047333034
```

4. Using the model from the previous question, the 5 unigrams/bigrams with the most positive associated weights and the 5 unigrams/bigrams with the most negative associated weights are:

```
The 5 unigrams/bigrams with the most positive associated weights are:
sort
a-bad
of-these
not-bad
the-best
The 5 unigrams/bigrams with the most negative associated weights are:
sort-of
water
corn
the-background
straw
```

5. The inverse document frequency of the words 'foam', 'smell', 'banana', 'lactic', and 'tart' are:

```
In the first review, idf score for  foam is 1.13786862069
In the first review, idf score for  smell is 0.537901618865
In the first review, idf score for  banana is 1.67778070527
In the first review, idf score for  lactic is 2.92081875395
In the first review, idf score for  tart is 1.80687540165
```

The tf-idf of the words 'foam', 'smell', 'banana', 'lactic', and 'tart' are:

```
In the first review, tf-idf score for  foam is 2.27573724138
In the first review, tf-idf score for  smell is 0.537901618865
In the first review, tf-idf score for  banana is 3.35556141054
In the first review, tf-idf score for  lactic is 5.8416375079
In the first review, tf-idf score for  tart is 1.80687540165
```

6. The cosine similarity between the first and the second review in terms of their tf-idf representations is (considering unigrams only):

```
Cosine similarity between the first and the second review is  0.0658819397474
```

7. Other review having the highest cosine similarity compared to the first review is:

```
Review having the highest cosine similarity compared to the first review is the one with beer ID 72146 and profile na
me spicelab
```

8. Adapt the original model that uses the 1000 most common unigrams, but replace the features with their 1000-dimensional tf-idf representations. The obtained MSE with the new model is:

```
The MSE is  0.278759560078
```

Appendix: (Some important parts)

Q1:

```
### Question 1
biCount = defaultdict(int)
punctuation = set(string.punctuation)
for d in data:
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
    words = r.lower().split()
    for i in range(len(words)-1):
        if not words[i]+'-'+words[i+1] in biCount:
            biCount[words[i]+'-'+words[i+1]] = 0
        biCount[words[i]+'-'+words[i+1]] += 1
print "There are ", len(biCount), "unique bigrams"
print"The 5 most frequently occuring bigrams are", heapq.nlargest(5, biCount, key=biCount.get)
```

Q2:

```
### Question 2
counts = [(biCount[w], w) for w in biCount]
counts.sort()
counts.reverse()

bigrams = [x[1] for x in counts[:1000]]

### Sentiment analysis
biId = dict(zip(bigrams, range(len(bigrams))))
biSet = set(bigrams)

def feature1(datum):
    feat = [0]*len(bigrams)
    r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
    words = r.split()
    for i in range(len(words)-1):
        if words[i]+'-'+words[i+1] in bigrams:
            feat[biId[words[i]+'-'+words[i+1]]] += 1
    feat.append(1) #offset
    return feat

X1 = [feature1(d) for d in data]
y = [d['review/overall'] for d in data]

#With regularization
clf = linear_model.Ridge(1.0, fit_intercept=False)
clf.fit(X1, y)
theta1 = clf.coef_
predictions1 = clf.predict(X1)

MSE1 = 0
for i in range(len(y)):
    MSE1 += (predictions1[i]-y[i])**2
    MSE1 /= len(y)
print "The MSE is ", MSE1
```

Q3:

```
### Question 3
wordCount = defaultdict(int)
for d in data:
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
    for w in r.split():
        wordCount[w] += 1

allwordCount = biCount.copy()
```

```
allwordCount.update(wordCount)
allcounts = [(allwordCount[w], w) for w in allwordCount]
allcounts.sort()
allcounts.reverse()
allgrams = [x[1] for x in allcounts[:1000]]

### Sentiment analysis
gramId = dict(zip(allgrams, range(len(allgrams))))
gramSet = set(allgrams)
```

```
X2 = [feature2(d) for d in data]y = [d['review/overall'] for d in data]

#With regularization
clf = linear_model.Ridge(1.0, fit_intercept=False)
clf.fit(X2, y)
theta2 = clf.coef_
predictions2 = clf.predict(X2)

MSE2 = 0
for i in range(len(y)):
    MSE2 += (predictions2[i]-y[i])**2
    MSE2 /= len(y)
print "The MSE is ", MSE2
```

Q4:

```
### Question 4
weights = [(theta2[i],allgrams[i]) for i in range(len(allgrams))]
weights.sort()
print "The 5 unigrams/bigrams with the most positive associated weights are:"
for i in range(5):
     print weights[999-i][1]

print "The 5 unigrams/bigrams with the most negative associated weights are:"
for i in range(5):
    print weights[i][1]
```

Q5:

```
### Question 5
tfwords = ['foam', 'smell', 'banana', 'lactic', 'tart']
idf = defaultdict(int)
tf = defaultdict(int)
for d in data:
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
    words = r.split()
    for w in tfwords:
        if w in words:
            idf[w] +=1

for w in idf:
    idf[w] = numpy.log10(5000/idf[w])

firstReview = data[0]['review/text']
r = ''.join([c for c in firstReview.lower() if not c in punctuation])
firstreviewWords = r.split()for w in firstreviewWords:
    if w in tfwords:
        tf[w] += 1

for w in idf:
    print "In the first review, idf score for ", w, "is", idf[w]
for w in tf:
    print "In the first review, tf-idf score for ", w, "is", tf[w] * idf[w]
```

Q6:

```
### Question 6
counts = [(wordCount[w], w) for w in wordCount]
counts.sort()
counts.reverse()
freq_words = [x[1] for x in counts[:1000]]

wordId = dict(zip(freq_words, range(len(freq_words))))
wordSet = set(freq_words)

idf = defaultdict(int)for d in data:
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
    words = r.lower().split()
    for w in wordSet:
        if w in words:
            idf[w] +=1

for w in idf:
    idf[w] = numpy.log10(5000/idf[w])

tfidf = {}for i in range(len(data)):
    tf = defaultdict(int)
    tfidf[i] = []
    r = ''.join([c for c in data[i]['review/text'].lower() if not c in punctuation])
    reviewWords = r.split()
    for w in reviewWords:
        tf[w] += 1
    for w in idf:
        tfidf[i].append(tf[w] * idf[w])

cosSim = 1 - spatial.distance.cosine(tfidf[0], tfidf[1])
print "Cosine similarity between the first and the second review is ", cosSim
```

Q7:

```
### Question 7
cosSet = [] for i in range(1,len(data)):
    sim = 1 - spatial.distance.cosine(tfidf[0], tfidf[i])
     cosSet.append(sim)
index = cosSet.index(cosSet == max)
print "Review having the highest cosine similarity compared to the first review is the one with beer ID", dat
a[index]['beer/beerId'], "and profile name", data[index]['user/profileName']
```

Q8:

```
### Question 8
X3=[]
for key, value in tfidf.iteritems():
    temp = value
    X3.append(temp)

for i in range(len(X3)):
    X3[i].append(1)

clf = linear_model.Ridge(1.0, fit_intercept=False)
clf.fit(X3, y)
theta3 = clf.coef predictions3 = clf.predict(X3)

MSE3 = 0
for i in range(len(y)):
     MSE3 += (predictions3[i]-y[i])**2
     MSE3 /= len(y)
print "The MSE is ", MSE3
```