HOMEWORK FOUR

Problem 1. Solution

The number of unique bigrams: 182246

Table 1:	The 5	most-freq	uently-oc	curring	bigrams:

Bigrams	The number of occurrences
'with a'	4587
'in the'	2595
'of the'	2245
'is a'	2056
'on the'	2033

Listing 1: Key code for Prob.1

```
1 wordCount = 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])
4
     w = r.split()
     for i in range(len(w)-1):
6
       wordCount[w[i]+' '+w[i+1]] += 1
   len(wordCount)
8
9
10
   counts = [(wordCount[w] , w) for w in wordCount]
11
   counts.sort()
   counts.reverse()
13
   counts
```

Problem 2. Solution

MSE using bigrams: 0.34315301406136378

Listing 2: Key code for Prob.2

```
words = [x[1] for x in counts[:1000]]
wordId = dict(zip(words, range(len(words))))
wordSet = set(words)
```

```
4
   def feature(datum):
     feat = [0]*len(words)
6
     r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
8
     w = r.split()
9
     for i in range(len(w)-1):
       bitemp=w[i]+' '+w[i+1]
10
11
       if bitemp in words:
12
         feat[wordId[bitemp]] += 1
13
     feat.append(1) #offset
14
     return feat
15
16 X = [feature(d) for d in data]
   y = [d['review/overall'] for d in data]
17
18
19
   clf = linear_model.Ridge(1.0, fit_intercept=False)
20 | clf.fit(X, y)
   theta = clf.coef_
   predictions = clf.predict(X)
   diff = predictions-y
24
   mse = sum([t**2 for t in diff])/len(diff)
  mse
```

Problem 3. Solution

MSE using unigrams and bigrams: 0.28904733303355806

Listing 3: Key code for Prob.3

```
1 wordCount = 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])
     w = r.split()
6
     for i in range(len(w)-1):
       wordCount[w[i]+' '+w[i+1]] += 1
8
     for tmp in w:
9
       wordCount[tmp] += 1
10 len(wordCount)
11
   counts = [(wordCount[w] , w) for w in wordCount]
12
13
   counts.sort()
   counts.reverse()
14
15
   # counts
16
   words = [x[1] \text{ for } x \text{ in counts}[:1000]]
17
   wordId = dict(zip(words, range(len(words))))
   wordSet = set(words)
18
19
20
   def feature(datum):
21
    feat = [0]*len(words)
22
     r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
     w = r.split()
23
24
     for i in range(len(w)-1):
25
       bitemp=w[i]+' '+w[i+1]
       if bitemp in words:
26
27
          feat[wordId[bitemp]] += 1
```

```
28
     for tmp in w:
29
       if tmp in words:
30
         feat[wordId[tmp]] += 1
31
     feat.append(1) #offset
32
     return feat
33
34 X = [feature(d) for d in data]
   y = [d['review/overall'] for d in data]
35
36
37
   clf = linear_model.Ridge(1.0, fit_intercept=False)
   clf.fit(X, y)
39
   theta = clf.coef_
   predictions = clf.predict(X)
41
42
   diff = predictions-y
   mse = sum([t**2 for t in diff])/len(diff)
```

Problem 4. Solution

Table 2: The 5 unigrams/bigrams with the most positive associated weights:

Unigrams/Bigrams	Weights
'sort'	0.51982780120456751
'a bad'	0.22881971426910591
'of these'	0.22283470424121538
'not bad'	0.21687721630732146
'the best'	0.20639109567227393

Table 3: The 5 unigrams/bigrams with the most negative associated weights:

Unigrams/Bigrams	Weights
'sort of'	-0.63976214971855316
'water'	-0.27048649882966247
'corn'	-0.23703101460442585
'the background'	-0.21624829959516467
'straw'	-0.19593772177944399

Listing 4: Key code for Prob.4

```
weight = [(theta[i] , words[i]) for i in range(len(words))]
weight.sort()
#weight

weight

weight.reverse()
#weight
```

Table 4: Inverse document frequency:

Words	Idf
'foam'	1.1378686206869628
'smell'	0.5379016188648442
'banana'	1.6777807052660807
'lactic'	2.9208187539523753
'tart'	1.8068754016455384

Table 5: Tf-idf Scores in the first review:

Words	Tf-idf
'foam'	2.2757372413739256
'smell'	0.5379016188648442
'banana'	3.3555614105321614
'lactic'	5.841637507904751
'tart'	1.8068754016455384

Listing 5: Key code for Prob.5

```
wordList = ['foam', 'smell', 'banana', 'lactic', 'tart']
   wordCount = defaultdict(int)
   punctuation = set(string.punctuation)
   for d in data:
5
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
7
     w = r.split()
8
     for word in wordList:
9
       if word in w:
10
         wordCount[word] += 1
11
12
   freq = [math.log10(len(data) * 1.0 /wordCount[word]) for word in wordList]
13
   #freq
14
   d=data[0]
15
16
   wordCount2 = defaultdict(int)
   r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
17
18
   w = r.split()
19
   for word in w:
20
       if word in wordList:
21
         wordCount2[word] += 1
22
   tfidf=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
23
24
   #tfidf
```

Problem 6. Solution

The cosine similarity between the rst and the second review in terms of their tf-idf representations: 0.106130241679

Listing 6: Key code for Prob.6

```
wordCount1 = defaultdict(int)
   punctuation = set(string.punctuation)
4
   for d in data:
     r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
5
6
     w = r.split()
7
     for word in wordList:
       if word in w:
         wordCount1[word] += 1
g
10
  freq = [math.log10(len(data) * 1.0 /wordCount1[word]) for word in wordList]
11
12
   d=data[0]
   wordCount2 = defaultdict(int)
14
   r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
15
   w = r.split()
16
   for word in w:
17
       if word in wordList:
18
         wordCount2[word] += 1
   tfidf1=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
19
20
21
   d=data[1]
   wordCount2 = defaultdict(int)
   r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
24
   w = r.split()
25
   for word in w:
26
        if word in wordList:
27
         wordCount2[word] += 1
   tfidf2=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
28
   1-scipy.spatial.distance.cosine(tfidf1,tfidf2)
```

Problem 7. Solution

Which other review has the highest cosine similarity compared to the rst review (provide the beerId and proleName, or the text of the review):

BeerId: 52211

User/profileName: 'Heatwave33'

Max cosine similarity: 0.31732766002633128

Number in the dataset: 4003

Listing 7: Key code for Prob.7

```
1
   cosine_m=0
   num_max = -1
3
4
   for t in range(1,len(data)):
       d=data[t]
6
       wordCount2 = defaultdict(int)
       r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
7
8
       w = r.split()
9
       for word in w:
10
          if word in wordList:
11
            wordCount2[word] += 1
12
        tfidf2=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
13
        tmp=1-scipy.spatial.distance.cosine(tfidf1,tfidf2)
14
       if (tmp>cosine_m):
15
          cosine_m=tmp
16
          num_max=t
17
   print data[num_max]
18
```

Problem 8. Solution

MSE: 0.27875956007772162

Listing 8: Key code for Prob.8

```
1 wordCount = 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])
4
5
     for w in r.split():
       wordCount[w] += 1
6
   counts = [(wordCount[w], w) for w in wordCount]
   counts.sort()
10
   counts.reverse()
11
12
   wordList = [x[1] for x in counts[:1000]]
13
14
   wordCount1 = defaultdict(int)
15
   punctuation = set(string.punctuation)
16
17
   for d in data:
18
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
19
     w = r.split()
20
     for word in wordList:
21
       if word in w:
22
         wordCount1[word] += 1
23
24
   freq = [math.log10(len(data) * 1.0 /wordCount1[word]) for word in wordList]
25
   wordId = dict(zip(words, range(len(words))))
27
   wordSet = set(words)
28
29
   def feature(datum):
```

```
feat = [0]*len(words)
31
     wordCount2 = defaultdict(int)
     r = ''.join([c for c in datum['review/text'].lower() if not c in punctuation])
     w = r.split()
34
     for word in w:
35
       if word in wordList:
36
         wordCount2[word] += 1
     for i in range(len(wordList)):
37
38
       feat[i] = freq[i] * wordCount2[wordList[i]]
39
     feat.append(1) #offset
40
     return feat
41
42 X = [feature(d) for d in data]
43
   y = [d['review/overall'] for d in data]
44
45 | clf = linear_model.Ridge(1.0, fit_intercept=False)
  clf.fit(X, y)
46
   theta = clf.coef_
48
   predictions = clf.predict(X)
49
50 diff = predictions-y
51 mse = sum([t**2 for t in diff])/len(diff)
```

Submitted by Zhang, Jinhan PID A53211930 on March 6, 2017.