Zhang, Jinhan PID A53211930
University of California, San Diego

March 6, 2017

HOMEWORK FOUR

Problem 1. Solution

The number of unique bigrams: 182246

Table 1: The 5 most-frequently-occurring 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

```
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()
  for i in range(len(w)-1):
    wordCount[w[i]+’ ’+w[i+1]] += 1
len(wordCount)

counts = [(wordCount[w] , w) for w in wordCount]
counts.sort()
counts.reverse()
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)
```

```

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

X = [feature(d) for d in data]
y = [d[’review/overall’] for d in data]

clf = linear_model.Ridge(1.0, fit_intercept=False)
clf.fit(X, y)
theta = clf.coef_
predictions = clf.predict(X)
diff = predictions-y
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

```
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()
  for i in range(len(w)-1):
    wordCount[w[i]+’ ’+w[i+1]] += 1
  for tmp in w:
    wordCount[tmp] += 1
len(wordCount)

counts = [(wordCount[w] , w) for w in wordCount]
counts.sort()
counts.reverse()
# counts
words = [x[1] for x in counts[:1000]]
wordId = dict(zip(words, range(len(words))))
wordSet = set(words)

def feature(datum):
  feat = [0]*len(words)
  r = ’’.join([c for c in datum[’review/text’].lower() if not c in punctuation])
  w = r.split()
  for i in range(len(w)-1):
    bitemp=w[i]+’ ’+w[i+1]
    if bitemp in words:
      feat[wordId[bitemp]] += 1
  for tmp in w:
    if tmp in words:
      feat[wordId[tmp]] += 1
  feat.append(1) #offset
  return feat

X = [feature(d) for d in data]
y = [d['review/overall'] for d in data]

clf = linear_model.Ridge(1.0, fit_intercept=False)
clf.fit(X, y)
theta = clf.coef_
predictions = clf.predict(X)

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.reverse()
#weight
```

Problem 5. Solution

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:
  r = ’’.join([c for c in d[’review/text’].lower() if not c in punctuation])
  w = r.split()
  for word in wordList:
    if word in w:
      wordCount[word] += 1

freq = [math.log10(len(data) * 1.0 /wordCount[word]) for word in wordList]
#freq

d=data[0]
wordCount2 = defaultdict(int)
r = ’’.join([c for c in d[’review/text’].lower() if not c in punctuation])
w = r.split()
for word in w:
    if word in wordList:
      wordCount2[word] += 1

tfidf=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
#tfidf
```

Problem 6. Solution

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

Listing 6: Key code for Prob.6

```python
wordCount1 = 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()
  for word in wordList:
    if word in w:
      wordCount1[word] += 1

freq = [math.log10(len(data) * 1.0 /wordCount1[word]) for word in wordList]
d=data[0]
wordCount2 = defaultdict(int)
r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
w = r.split()
for word in w:
    if word in wordList:
      wordCount2[word] += 1
tfidf1=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]

d=data[1]
wordCount2 = defaultdict(int)
r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
w = r.split()
for word in w:
    if word in wordList:
      wordCount2[word] += 1
tfidf2=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
1-scipy.spatial.distance.cosine(tfidf1,tfidf2)
```

Problem 7. Solution

Which other review has the highest cosine similarity compared to the first review (provide the beerId and profileName, 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

```
cosine_m=0
num_max=-1

for t in range(1,len(data)):
    d=data[t]
    wordCount2 = defaultdict(int)
    r = ''.join([c for c in d['review/text'].lower() if not c in punctuation])
    w = r.split()
    for word in w:
      if word in wordList:
        wordCount2[word] += 1
    tfidf2=[freq[i]*wordCount2[wordList[i]] for i in range(len(wordList))]
    tmp=1-scipy.spatial.distance.cosine(tfidf1,tfidf2)
    if(tmp>cosine_m):
      cosine_m=tmp
      num_max=t

print data[num_max]
```

Problem 8. Solution

MSE: 0.27875956007772162

Listing 8: Key code for Prob.8

```
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])
  for w in r.split():
    wordCount[w] += 1

counts = [(wordCount[w], w) for w in wordCount]
counts.sort()
counts.reverse()

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

wordCount1 = 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()
  for word in wordList:
    if word in w:
      wordCount1[word] += 1

freq = [math.log10(len(data) * 1.0 /wordCount1[word]) for word in wordList]

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

def feature(datum):
  feat = [0]*len(words)
  wordCount2 = defaultdict(int)
  r = ’’.join([c for c in datum[’review/text’].lower() if not c in punctuation])
  w = r.split()
  for word in w:
    if word in wordList:
      wordCount2[word] += 1
  for i in range(len(wordList)):
    feat[i]=freq[i]*wordCount2[wordList[i]]
  feat.append(1) #offset
  return feat

X = [feature(d) for d in data]
y = [d[’review/overall’] for d in data]

clf = linear_model.Ridge(1.0, fit_intercept=False)
clf.fit(X, y)
theta = clf.coef_
predictions = clf.predict(X)

diff = predictions-y
mse = sum([t**2 for t in diff])/len(diff)
```